Aalto University
School of Engineering
Department of Applied Mechanics

Jaro Hokkanen

Introduction of a segment-to-segment penalty contact formulation

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Technology

Espoo, January 24, 2014

Supervisor: Professor Olli Saarela

Thesis advisor(s): Dr. Guido Dhondt

Aalto University
School of Engineering

Aalto University
P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author: Jaro Hokkanen	
Title of thesis: Introduction of a segment-to-segment penalty contact formulation	
Degree programme: Mechanical Engineering	
Department: Department of Applied Mechanics	
Professorship: Aeronautical Engineering	**Code:** Kul-34
Thesis supervisor: Professor Olli Saarela	
Thesis advisor(s): Dr. Guido Dhondt	

Abstract

The modeling of contact problems in solid mechanics using the finite element method is a challenging and complicated task. Stable, efficient and accurate algorithms are required for finding an effective solution for general contact problems. This thesis introduces a transition from a node-to-segment penalty contact formulation to an effective segment-to-segment penalty contact formulation.

The main issues of the node-to-segment approach are convergence problems and inaccurate results in case of nonconforming meshes. These disadvantages are caused by the fact that the contact constraints are satisfied only at the nodal locations. The relatively new segment-to-segment formulations provide a way to apply the constraint conditions along the entire boundary in a weak integral sense. This usually results in better stability and accuracy.

Several different discretization schemes for the proposed segment-to-segment formulation are presented. Moreover, additional complexities caused by the new formulation are discussed and the solutions for these problems are introduced. The different discretization schemes are compared by simple examples as well as big and complicated real models. Additionally, the widely accepted numerical benchmark test known as “patch test” is conducted to compare the results given by the different contact implementations.

The results obtained by the comparisons support the expected outcome of the segment-to-segment approach. The advantages of the segment-to-segment formulations become clear and a much better contact algorithm is introduced into an open source finite element analysis program.

Date: 24.01.2014	**Number of pages:** 8 + 113
Keywords: contact, penalty method, node-to-segment, segment-to-segment, patch test	

Acknowledgements

This study has been completed for MTU Aero Engines AG, during the period of July 15th, 2013 – January 14th, 2014, in Munich, Germany.

I would like to express my very great appreciation to Dr. Guido Dhondt, the instructor of this work, for this great opportunity and the guidance he has provided to me. I would also like to offer my special thanks to my supervisor Professor Olli Saarela for all the help and careful revision of the thesis.

I wish to acknowledge the help provided by my colleagues and friends during my stay in Munich. I am also very grateful for the support provided by my parents and siblings. Finally, I wish to extend my warmest embrace to Niina Salminen to express my gratitude for always being there and supporting me throughout my studies.

Espoo, January 24, 2014

Jaro Hokkanen

Table of Contents

Abbreviations

3D	Three-dimensional
CCX	CalculiX CrunchiX
CGX	CalculiX GraphiX
CFD	Computational Fluid Dynamics
GC7	Gauss Cut formulation (seven point scheme)
CPU	Central Processing Unit
LGC7	Linear Gauss Cut formulation (seven point scheme)
MPC	Multiple Point Constraint
Nodes	Node-to-segment formulation (CalculiX)
NTS	Node-to-segment formulation (Abaqus)
STS	Segment-to-segment formulation (Abaqus)

Symbols

a	coefficient of a polynomial
A	general matrix
b	coefficient of a polynomial
c	coefficient of a polynomial
C	number of contact elements
c_0	maximum clearance for a contact element to be created
c_p	specific heat
d	overclosure / coefficient of a polynomial
D	auxiliary variable
E	elastic modulus
f	random function / contact force depending on the clearance
F	force
F	normal contact force
h	film coefficient
i	index number / number of integration points
I	identity matrix
K	spring coefficient (penalty parameter)
m	mass
m	normal vector of the master surface
n	unit vector of **m**
p	pressure / integration point
p	location of an integration point
p_0	pressure at zero clearance
q	projection of **p** onto the master surface
$\mathbf{q}_j$	locations of the master nodes
R	absolute value of a residual force
S	general matrix
t	general variable
u	displacement vector
x	general variable
y	general variable
z	general variable
β	angle / coefficient depending on c_0
δ_{ij}	Kronecker delta function
ϵ	number depending on σ_∞ and K
ϵ_N	number determining the accuracy of the Newton-Raphson method
η	local coordinate
Λ	tangent of the stick range

λ	clearance
$\boldsymbol{\lambda}$	vector between **q** and **p**
λ_c	conductivity
μ	friction coefficient
μ_v	dynamic viscosity
ξ	local coordinate
ρ	density
σ_∞	value for tension at large clearances
φ	shape function
Φ	penalty function

1 Introduction

The numerical analysis of structural contact problems has been an important subject of interest over the past decades. Despite the significant progress achieved in the subject, it is still one of the most difficult aspects of nonlinear structural analysis (Weyler et al., 2012). The purpose of this master`s thesis is to develop a better contact algorithm to an open source finite element analysis program in terms of accuracy and stability.

1.1 Contact

While developing a new contact formulation, the first thing is to decide how the contact constraints are enforced. The most common techniques are the penalty methods and the Lagrange multiplier methods, but also other techniques such as the augmented Lagrangian approach or the relatively new Nitsche's method are proposed in the literature. (Weyler et al., 2012)

Penalty methods enforce the contact conditions by introducing a parameter of large value into the governing equations of equilibrium. This penalty parameter acts like a stiff spring between the contact surfaces. In theory, as the penalty parameter tends to infinity, the constraint condition is satisfied exactly. However, the resulting system of equations may become ill-conditioned if the penalty parameter is too large. (Baig, 2006; Weyler et al., 2012)

In the Lagrange multiplier method (also known as mortar method), additional auxiliary variables (the Lagrange multipliers) are introduced. The method enforces the contact constraints exactly, but the degrees of freedom are increased by the number of auxiliary variables which leads to a substantially big coefficient matrix and greater computational effort. (Baig, 2006; Weyler et al., 2012)

Augmented Lagrange method combines the Lagrange and the penalty approaches by having two sets of contact forces; the Lagrange multipliers and the penalty forces. This arrangement provides the regularizing effect of the penalty method and ensures the exact satisfaction of the constraints by the Lagrange multiplier method, without having the ill-conditioning problem of the traditional penalty method. (Zavarise, 1999)

The method of Nitsche treats the boundary or the interface conditions in a weak sense by applying a consistent penalty term. In contrast to the Lagrange multiplier methods, no additional unknowns (Lagrange multipliers) are needed and thus additional degrees of freedom are avoided. Moreover, the proposed method is variationally consistent unlike the typical penalty methods. (Chouly & Hild, 2013; Chouly et al., 2013; Annavarapu, 2014)

Another important topic is the discretization scheme of the contact interface. The different types of contact interface treatments used in many finite element applications include node-to-node, node-to-segment and segment-to-segment formulations. However, the current trend is shifting towards segment-to-segment methods due to their many incontestable advantages.

In the node-to-node approach the non-penetrating condition is applied for the opposite nodes, but a one-to-one correspondence between the boundary nodes on the contacting surfaces are required. This method is not suitable if the meshes of the contact surfaces are nonconforming or large displacements are involved. The advantage of this method is its obvious algorithmic simplicity and unbiased treatment of the contact surfaces. However, the node-to-node formulations are not commonly used nowadays due to their limitations. (Taylor & Papadopoulos 1991)

Node-to-segment formulations enforce the non-penetration conditions such that the nodes of the dependent surface (slave) are prevented from penetrating the opposite independent surface (master). This is also known as a “single-pass” algorithm, which always results in a biased treatment of contact surfaces, depending on the choice of the slave and the master surface. The geometric bias can however be eliminated by reversing the roles of the surfaces and repeating the same process. This is known as a “two-pass” algorithm. The disadvantage of the two-pass algorithm is that it is prone to lock due to the overconstraining of the contact surface. For node-to-segment formulations, the meshes do not need to be conforming and large displacements can be treated. However, disadvantages such as failure to pass the patch test and possible convergence problems are often encountered. The node-to-segment approach initially

developed by Hallquist et al. (1985) is commonly used in many finite element applications. (Taylor & Papadopoulos 1991; Weyler et al., 2012)

Relatively new segment-to-segment discretization schemes apply the constraint conditions along the entire boundary in a weak integral sense. The segment-to-segment formulations may result in a biased or unbiased treatment of the contact surfaces depending on the applied discretization scheme. The method is well suited to exchange information between two non-conforming surface grids. The segment-to-segment algorithms typically pass the patch test provided that the integration point scheme is dense enough. The convergence rate of the segment-to-segment formulations is also usually significantly better compared to node-to-segment formulations. The segment-to-segment approach has lately found its way in many of the recently developed large deformation problems. (Puso & Laursen 2004; Weyler et al., 2012)

1.2 Problem formulation

The programming environment for this master`s thesis is the open source finite element analysis program CalculiX. At the moment of writing, the contact modeling in CalculiX is based on a node-to-segment penalty method or a segment-to-segment mortar method (Lagrange multiplier method).

The main drawback of the current node-to-segment formulation is its relatively poor stability. The method fails to converge for many more complicated models. Additionally, the node-to-segment algorithm suffers from accuracy problems in case of nonconforming meshes. The mortar algorithm instead converges better, but at the time of writing it is still under development and also has controversies being included in the open source program.

The main objective of this master`s thesis is to extend the current node-to-segment algorithm into a segment-to-segment version. This contact formulation will be based on the traditional penalty method due to its easy interpretation and relatively simple implementation. Different discretization schemes are proposed and compared to find the most effective solution. Furthermore, a linearized approach is presented and discussed. The stability of the proposed method is of the greatest importance, but also better accuracy is expected.

Moreover, additional complexities caused by the new formulation are expected to be solved within the timeframe of the master`s thesis. The expected outcome is such that the new functional and tested version of CalculiX can be officially released shortly after the assignment is finished.

1.3 Thesis outline

In the chapter 2, the programming environment, CalculiX, is presented. The solver program, but also the pre- and postprocessor are briefly covered. Additionally, the unit system, basic three-dimensional elements and the concept of shape functions used in CalculiX are discussed.

The chapter 3 introduces the contact formulations already implemented in CalculiX and the procedure to solve nonlinear problems. Additionally, the general contact formulation for the node-to-segment penalty method is derived and the disadvantages of this formulation are discussed.

The chapter 4 presents the modifications required for the transition from the node-to-segment formulation to a segment-to-segment formulation. Moreover, the different discretization schemes are introduced. The problems encountered by the new approach are solved and the linear version is discussed.

In the chapter 5, a comparison between different contact formulations is performed for the simple examples. For these examples, the boundary conditions and the loadings are easy to understand. The comparison for the accuracy and the stability is conducted between Abaqus and different contact formulations implemented in CalculiX. Moreover, the widely known contact patch test is performed.

The chapter 6 compares the stability and the accuracy of the results for the complex real models related to turbine engine parts. The results are mainly compared to Abaqus, because it is expected to give reliable results.

Finally in the chapter 7 the new contact formulations are discussed and the pros and the cons are weighted. The influence of the linearized version is also gone through. Suggestions for further development of the program are given. At the very end, a comprehensive conclusion about the master`s thesis work is presented.

2 CalculiX

CalculiX is an open source finite element analysis program, which is composed of a pre- and postprocessor CalculiX GraphiX (CGX) and an implicit and explicit solver CalculiX CrunchiX (CCX). The program is mainly developed by Guido Dhondt (CCX) and Klaus Wittig (CGX). The original version is written for Linux, but the program has also been ported to the Windows operating system at the later stage.

Finite element analysis is a practical application of the finite element method. The idea is to divide a complex problem into small elements, which is called mesh generation. Finite element analysis can be divided into three separate stages. These stages are presented in the figure 1.

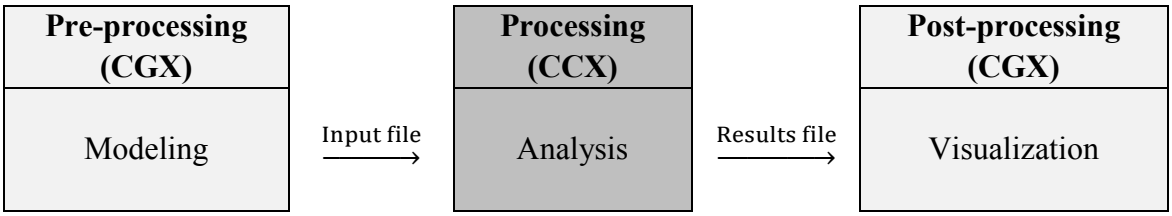


Figure 1. The stages of finite element analysis.

2.1 CalculiX GraphiX

CalculiX GraphiX (CGX) is a program for pre- and post-processing finite element models. It is designed to generate and display the finite element model as well as the results calculated by a solver program. The program is mainly controlled by the keyboard with different commands for each function. The mouse is mainly used for moving the model, but there is also a pop-up menu for the commands which are considered the most important.

CalculiX GraphiX can be used to generate and display beam, shell and brick elements in a linear and quadratic form. Pentahedral- and tetrahedral -elements can be displayed but not generated. A structured mesh is created based on a description of the corresponding geometry. After creating the mesh the information is written to a file for further use with the solver program. The pre-processor can generate input data for the commercial finite element method programs such as Nastran, Ansys and Abaqus. It is also able to generate grid data for the Computational Fluid Dynamics programs duns, ISAAC and OpenFOAM. (Wittig, 2013) Von Mises stress of a turbine segment visualized by CalculiX GraphiX is shown below:

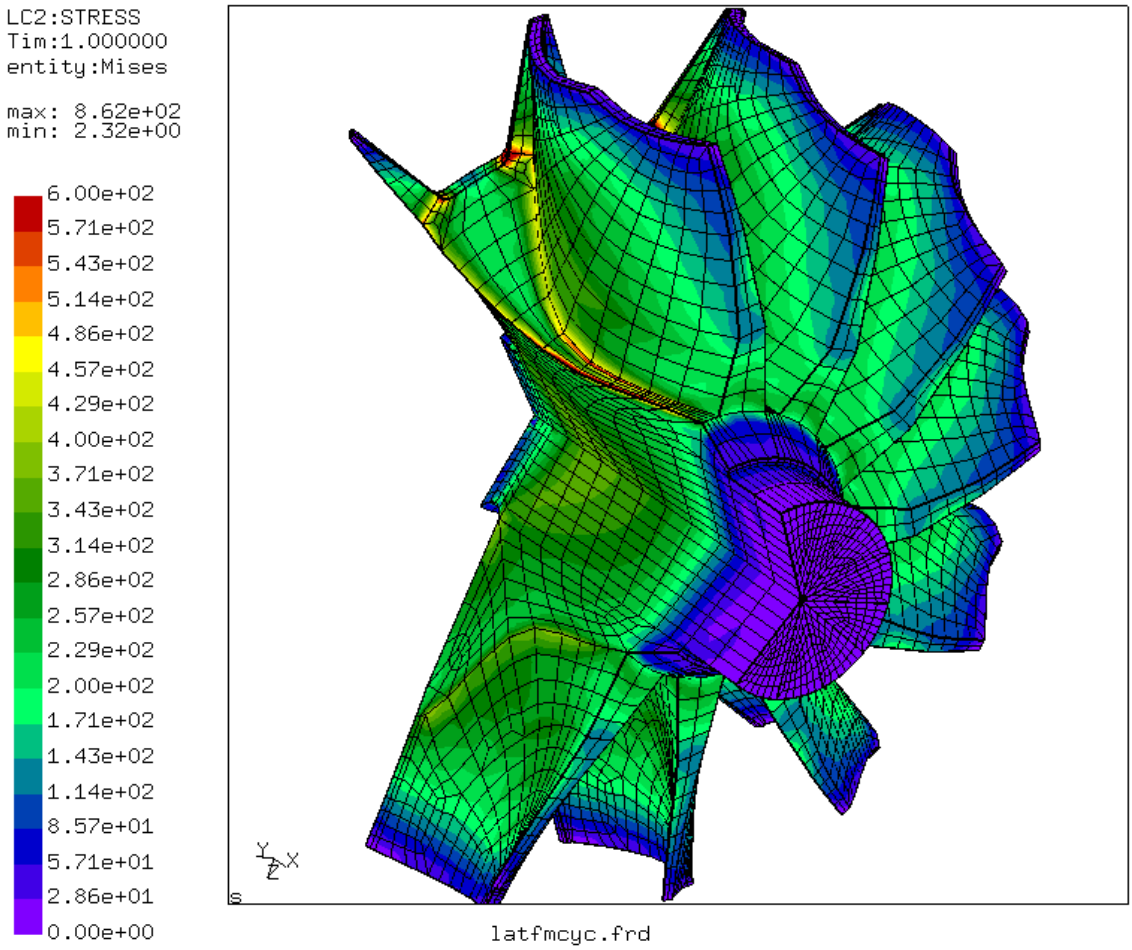


Figure 2. Von Mises stress in a turbine segment (Dhondt, [no date]).

2.2 CalculiX CrunchiX

CalculiX CrunchiX (CCX) is the solver program, which is able to do linear and non-linear calculations. It supports multiple types of analyses to be performed. For example static, dynamic and thermomechanical analysis, as well as frequency-, buckling- and CFD-analyses are possible. Static or dynamic problems may contain nonlinearities, which are caused by geometry, material behavior or contact. Geometric nonlinearities may be caused by large deformations, whereas plasticity of the material also leads to nonlinear calculations. CalculiX CrunchiX uses Abaqus input-file format (inp-file) and stores the results of the calculation into an frd-file. The program supports multiple CPU's, which allows calculations to be performed in a parallel way. (Dhondt, 2013)

CalculiX CrunchiX is written in FORTRAN and C. C is used mainly for automatic allocation and reallocation purposes. Calculations are mostly performed in FORTRAN subroutines.

CalculiX is not aware of units. The units are determined by the user, when the input data is written. The units can be freely chosen by the user, provided that the numbers have consistent units. The table 1 gives an overview about three different systems of units.

Table 1. Suggestions for different possible unit systems. (Dhondt, 2013)

Symbol	Meaning	System of units		
		m, kg, s, K	**mm, g, s, K**	**mm, N, s, K**
E	Young´s modulus	$1\frac{\mathrm{kg}}{\mathrm{ms}^2}$	$1\frac{\mathrm{g}}{\mathrm{mms}^2}$	$10^{-6}\frac{\mathrm{N}}{\mathrm{mm}^2}$
ρ	Density	$1\frac{\mathrm{kg}}{\mathrm{m}^3}$	$10^{-6}\frac{\mathrm{g}}{\mathrm{mm}^3}$	$10^{-12}\frac{\mathrm{Ns}^2}{\mathrm{mm}^4}$
F	Force	$1\frac{\mathrm{kgm}}{\mathrm{s}^2}$	$10^{6}\frac{\mathrm{gmm}}{\mathrm{s}^2}$	1N
m	Mass	1kg	1g	$10^{-3}\frac{\mathrm{Ns}^2}{\mathrm{mm}}$
c_p	Specific heat	$1\frac{\mathrm{m}^2}{\mathrm{s}^2\mathrm{K}}$	$10^{6}\frac{\mathrm{mm}^2}{\mathrm{s}^2\mathrm{K}}$	$10^{6}\frac{\mathrm{mm}^2}{\mathrm{s}^2\mathrm{K}}$
λ_c	Conductivity	$1\frac{\mathrm{kgm}}{\mathrm{s}^3\mathrm{K}}$	$10^{6}\frac{\mathrm{gmm}}{\mathrm{s}^3\mathrm{K}}$	$1\frac{\mathrm{N}}{\mathrm{sK}}$
h	Film coefficient	$1\frac{\mathrm{kg}}{\mathrm{s}^3\mathrm{K}}$	$10^{3}\frac{\mathrm{g}}{\mathrm{s}^3\mathrm{K}}$	$10^{-3}\frac{\mathrm{N}}{\mathrm{mmsK}}$
μ_v	Dynamic viscosity	$1\frac{\mathrm{kg}}{\mathrm{ms}}$	$1\frac{\mathrm{g}}{\mathrm{mms}}$	$10^{-6}\frac{\mathrm{Ns}}{\mathrm{mm}^2}$

2.3 Elements

In CalculiX, one- and two-dimensional elements are expanded into three-dimensional elements. A short overview about the basic 3D elements used in CalculiX is provided in this section. The naming of the elements follows the naming used in Abaqus. General purpose solid elements are called C3Dx and corresponding fluid elements F3Dx.

2.3.1 Brick elements

The **C3D8** linear brick element consists of eight nodes and eight integration points. The node numbering and the integration point scheme are shown in the figures 3 and 4. Due to the full integration, shear or volumetric locking might occur with this element type. (Dhondt, 2013)

The **C3D8R** linear element is similar to the C3D8 element, but with reduced integration. It consists of only one integration point located in the middle of the element. Due to the reduced integration, the locking phenomenon with the C3D8 is avoided. However, inaccurate results might occur due to the presence of only one integration point. Additionally, so called hourglassing problem is present leading to incorrect displacements. (Dhondt, 2013)

The **C3D8I** incompatible mode element is also similar to the C3D8, but the shear locking problem is removed and the volumetric locking phenomenon has been reduced.

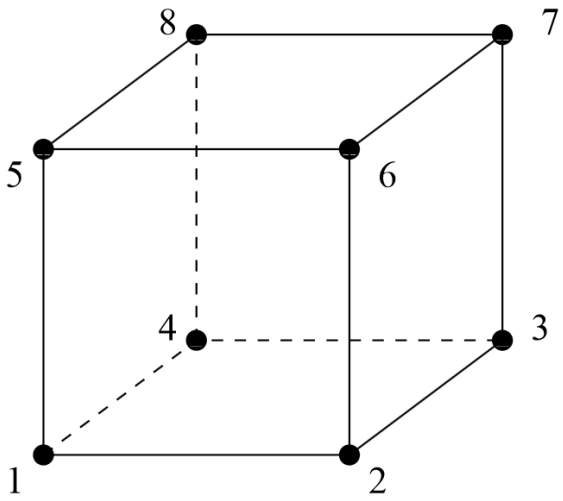


Figure 3. An eight-node brick element. (Dhondt, 2013)

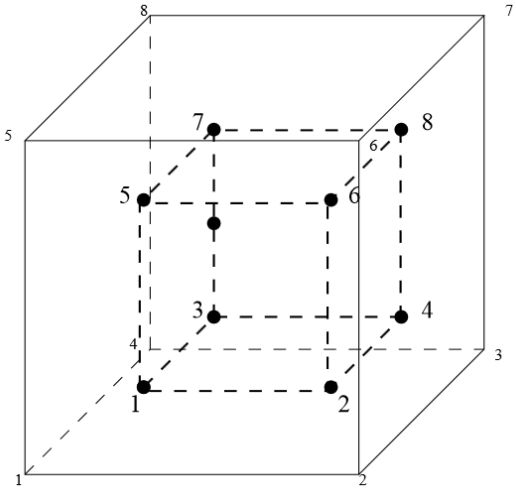


Figure 4. The integration point scheme for C3D8 and C3D20R elements. (Dhondt, 2013)

The **C3D20** is a quadratic brick element with twenty nodes. This element is much more accurate than the C3D8 element, but it still exhibits the same problems with locking phenomena, but in the much smaller scale. The node numbering and the integration point scheme are shown in the figures 5 and 6.

The **C3D20R** is a quadratic brick element with reduced integration. It consists of eight integration points, which are located corresponding to the figure 4. The hourglassing problem is rare with this element type.

The **C3D20RI** element is similar to the C3D20R element, but the isochoric condition is applied at the corner nodes of the element.

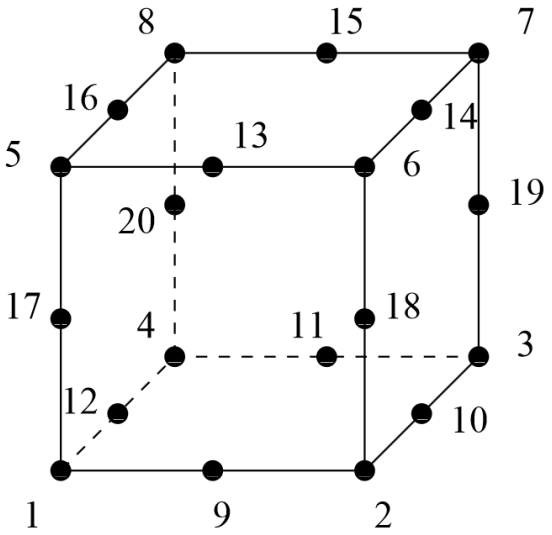


Figure 5. A 20-node brick element. (Dhondt, 2013)

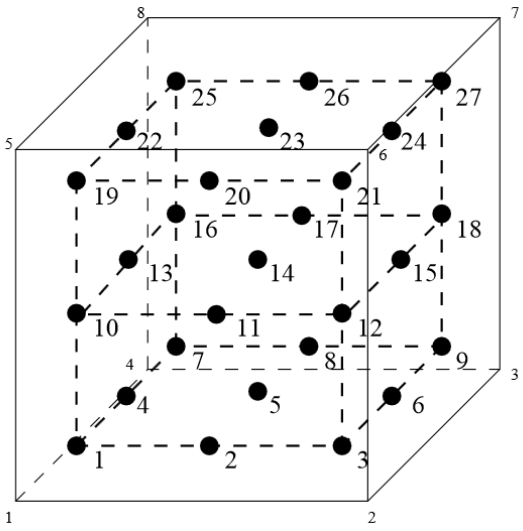


Figure 6. The integration point scheme for the C3D20 element. (Dhondt, 2013)

2.3.2 Tetrahedral elements

The **C3D4** is a linear tetrahedral element with four nodes and one integration point. This element is not usually recommended for structural calculations unless the mesh is very dense. The element and the node numbering correspond to the figure 7, with only the corner nodes existing.

The **C3D10** is a quadratic tetrahedral element with ten nodes and four integration points. The behavior of the element is good. The element and the node numbering are shown in the figure 7.

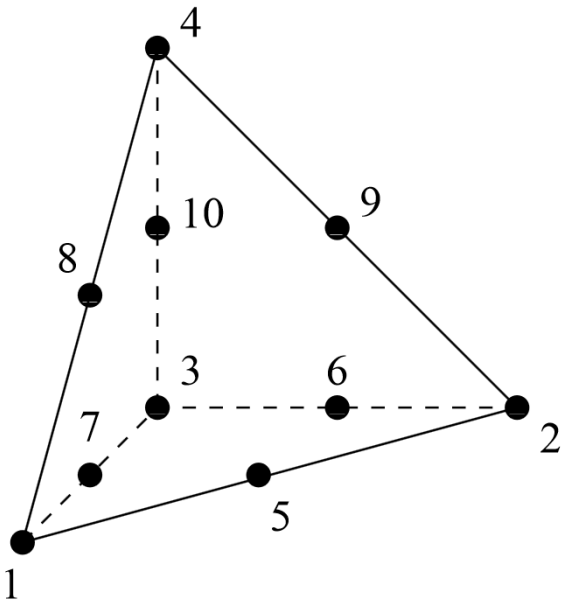


Figure 7. A ten-node tetrahedral element. (Dhondt, 2013)

2.3.3 Wedge elements

The **C3D6** is a linear wedge element with six nodes and two integration points. It is not recommended for structural calculations unless the mesh is very dense. The element and the node numbering correspond to the figure 8, with only the corner nodes existing.

The **C3D15** is a quadratic wedge element with 15 nodes and nine integration points. The behavior of the element is good. The element and the node numbering are shown in the figure 8.

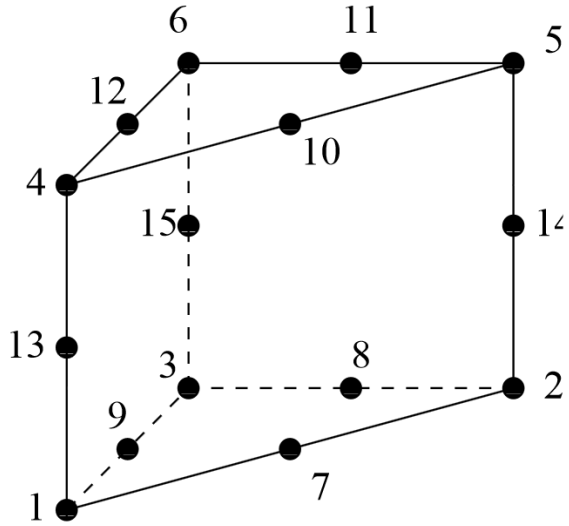


Figure 8. A 15-node wedge element. (Dhondt, 2013)

2.4 Shape functions

The shape function is a function which interpolates the solution between the discrete values obtained at the nodes. Typically low order polynomials are used as the shape functions. In CalculiX, the same shape functions are used for the displacements and the geometry. The shape functions for two-dimensional faces with four nodes are derived below.

The local coordinates $\xi \text{and} \, \eta$ are determined such that $-1 \leq \xi, \eta \leq 1$. The following requirement is satisfied for the shape functions:

$$\varphi_i(\xi_j, \eta_j) = \delta_{ij}, \tag{1}$$

where δ_{ij} is Kronecker delta function. It is defined as follows

$$\delta_{ij} = \begin{cases} 0 & if \quad i \neq j \\ 1 & if \quad i = j \end{cases} \tag{2}$$

The lowest-order polynomial with four unknown coefficients has the form:

$$\varphi_i(\xi_j, \eta_j) = a_i + b_i\xi + c_i\eta + d_i\xi\eta \tag{3}$$

When the equations 1-3 are combined, there are four equations and four unknowns. After solving these functions, one is left with the following shape functions.

$$\begin{aligned} \varphi_1 &= (1-\xi)(1-\eta)/4 \\ \varphi_2 &= (1+\xi)(1-\eta)/4 \\ \varphi_3 &= (1+\xi)(1+\eta)/4 \\ \varphi_4 &= (1-\xi)(1+\eta)/4 \end{aligned} \tag{4}$$

Other shape functions are not presented in this thesis, but they are derived similarly. The shape functions for the basic three dimensional elements can be found from literature. (Dhondt, 2004)

2.5 Re-meshing

Re-meshing is used to transform the underlying quadratic elements of the contact surfaces into linear ones. At the moment of writing all independent contact surfaces in CalculiX are re-meshed for the node-to-segment penalty method in case the contact surfaces consist of quadratic elements. The dependent surfaces are also re-meshed if they are defined based on faces and are therefore not nodal surfaces. Re-meshing is conducted due to the problems in the contact calculation caused by quadratic surfaces. Quadratic elements are re-meshed into C3D8I, C3D4 or C3D6 depending whether they are hexahedral, tetrahedral or wedge type elements. The picture of re-meshed contact surfaces of the simple cube example is shown in the figure 9. The original mesh consists of C3D20 elements, and both contact surfaces are re-meshed into linear C3D8I elements.

However, re-meshing makes the analysis more complicated and causes discontinuous results at the contact surfaces for the current node-to-segment penalty contact formulation in CalculiX. Even though the problem with the results could be fixed, it was noticed that the convergence problems encountered earlier with original node-to-segment penalty version did not exist with the new linear segment-to-segment penalty implementation (chapter 4). Therefore there is no need for re-meshing for linear penalty version, since the convergence does not seem to suffer at all, and the results at the contact surfaces are much better.

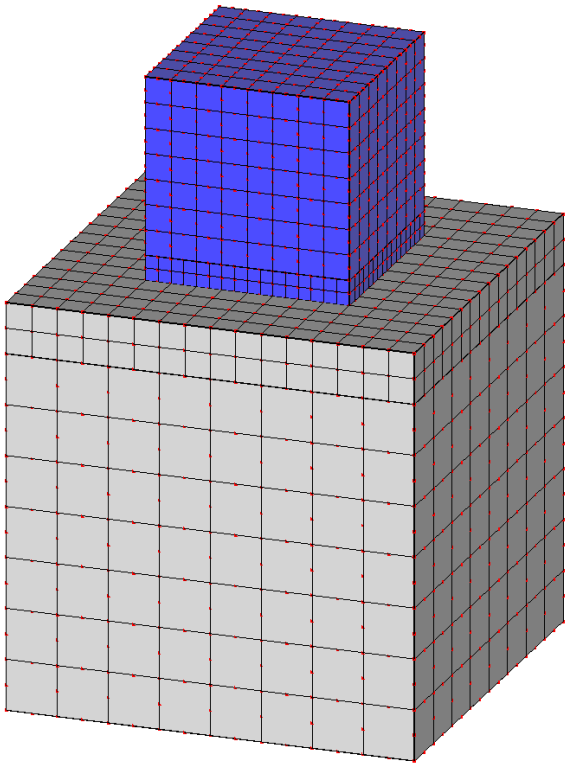


Figure 9. Re-meshing of the contact surfaces. The red dots represent nodes.

3 Contact modeling

In this chapter, different contact definitions already implemented in CalculiX are presented. At the moment of writing, the definition of contact in the official version of CalculiX was based on the node-to-segment penalty method (Nodes) or the segment-to-segment mortar method. Both methods are based on a pairwise interaction of the corresponding surfaces, and cannot be mixed in the same input deck. The pair of contacting surfaces consists of a slave (dependent surface) and a master surface (independent surface). Contact pairs are defined in the input deck and the amount of contact pairs is unlimited.

In the penalty method, a large positive term called penalty parameter "penalizes" the dissatisfaction of the constraint condition. In theory, as the penalty parameter tends to infinity, the constraint condition is satisfied exactly. However, the resulting system of equations may become ill-conditioned when the penalty parameter is too large. This means that the penalty method satisfies the constraint conditions in an approximate sense depending on the choice of the penalty parameter. (Baig, 2006; Weyler et al., 2012)

The mortar method (also called Lagrange multiplier method) introduces an additional variable called Lagrange multiplier to the constrained optimization problem, after which the gradient of the resulting function (also called Lagrangian) is set to zero. When

Lagrange multipliers are used to solve constrained optimization problems, additional Kuhn-Tucker conditions must be satisfied to ensure the meaningful existence of the Lagrange multipliers, i.e. the contact force is pushing the contacting objects away from each other. Furthermore, the contact is called “hard contact”, because the constraint condition is satisfied exactly. In the mortar method, the coefficient matrix becomes substantially big and the computational effort required may be quite large. (Gu et al., 2002; Weyler et al., 2012)

3.1 Nonlinear calculations

Discontinuities in the structure due to a contact cause nonlinear behavior. In CalculiX, nonlinear problems are solved by the Newton-Raphson method. For nonlinear static analysis, which is considered in this thesis, the loads are applied in time increments. These increments represent a pseudo time, which denotes the intensity of the applied loads at a certain step. The initial time increment is chosen by the user. Subsequent increments are controlled by the automatic incrementation control, but also fixed increment length can be chosen. Within each increment, the program iterates until the equilibrium between the internal structure forces and the externally applied loads is reached. If the convergence is too slow, or divergence occurs, a new attempt is made with a smaller increment length. This is illustrated in the figure 10.

In the Newton-Raphson method, the problem is broken into linear parts and solved by iteration. For illustration of the method, a one-dimensional nonlinear function is considered.

$$f(x) = 0 \tag{5}$$

To find the solution to this problem, there must be an initial guess, x_0, for the solution. The function $f(x_0)$ is now replaced by its tangent at the point x_0. Now, the equation becomes

$$f(x_0) + (x_1 - x_0)f'(x_0) = 0, \tag{6}$$

which is now a linear equation. The solution of this equation yields an approximation of the solution, which is called point x_1:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{7}$$

The aforementioned procedure is conducted subsequently for x_1, to achieve the next more accurate approximate solution. The formula producing x_{n+1} from x_n is expressed as follows:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \tag{8}$$

and is known as the Newton-Raphson formula. The principle of the Newton-Raphson method is shown in the figure 11.

The procedure is continued until the relative difference between the two consecutive solutions is smaller than a specified value ϵ_N:

$$\left|\frac{x_n - x_{n-1}}{x_{n-1}}\right| \leq \epsilon_N \tag{9}$$

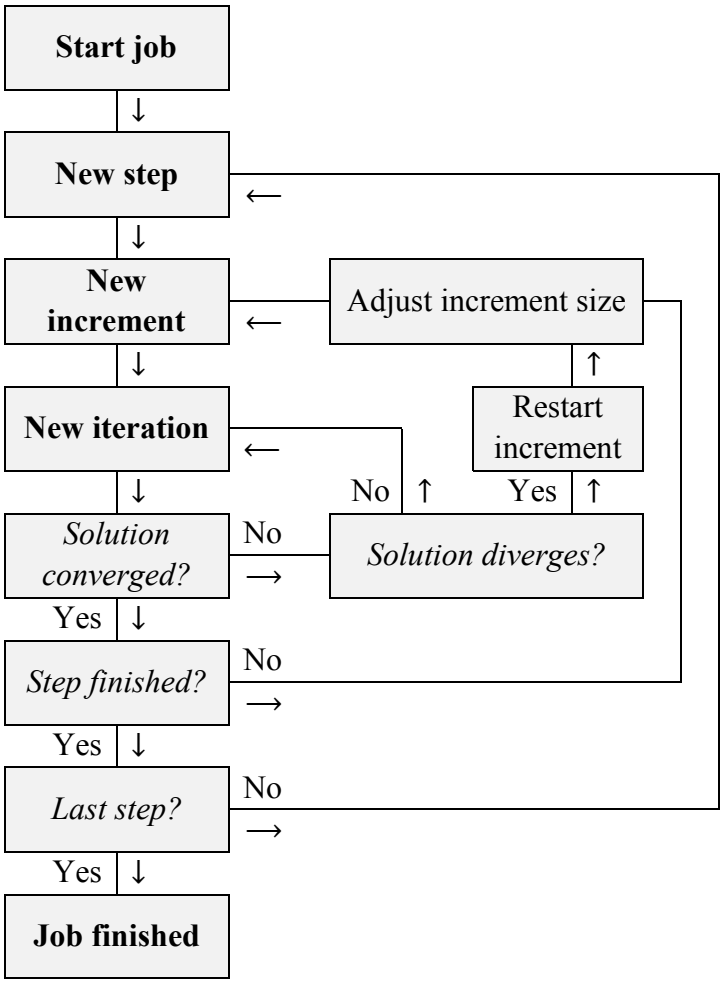


Figure 10. The principle of a nonlinear calculation.

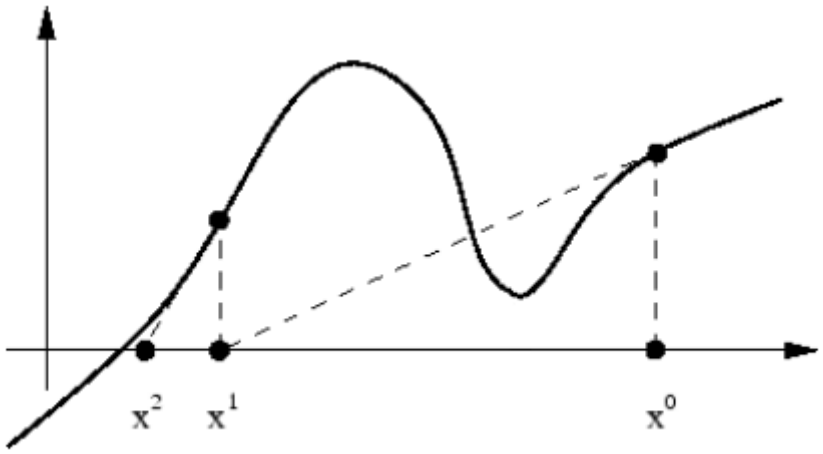


Figure 11. The principle of the Newton-Raphson method. (Jahn, 2007)

The convergence of the Newton-Raphson method depends on how close the initial guess is to the real solution. The smoothness of the function is also an important factor. If there is a local maximum or a minimum between the final real solution and the initial guess, there may be no convergence. This behavior is shown in the figure 12, where the method is stuck in an infinite cycle at the local minimum. The method also fails to converge if a stationary point of the function is encountered i.e. the derivative of the function is zero. Moreover, derivatives which are not zero but still close to zero cause convergence problems (figure 13). Additional convergence problems are also caused by asymptotic behavior (figure 14) and numerical overflow (figure 15).

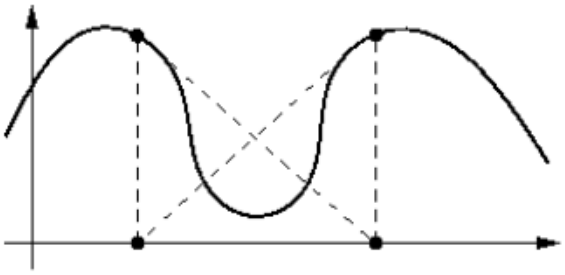


Figure 12. Local minimum and oscillating behavior. (Jahn, 2007)

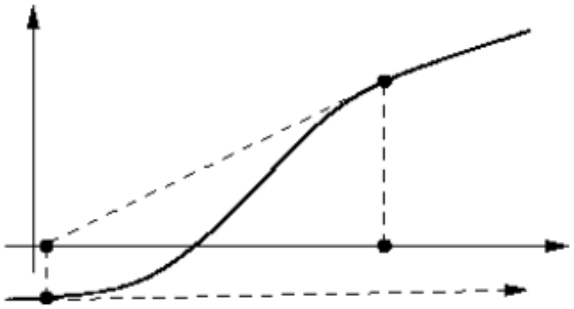


Figure 13. Derivative close to zero. (Jahn, 2007)

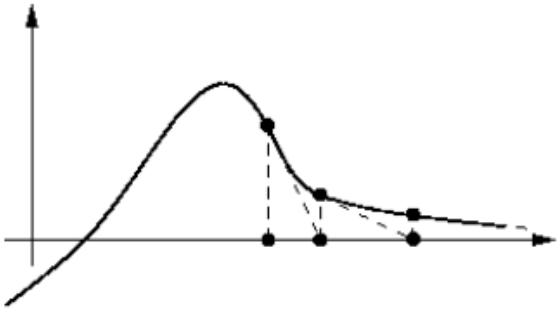


Figure 14. Asymptotic behavior. (Jahn, 2007)

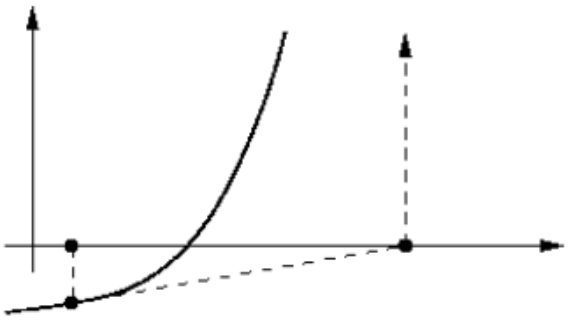


Figure 15. Numerical overflow. (Jahn, 2007)

3.2 Penalty contact

Penalty methods are certain algorithms used to solve constrained optimization problems. A penalty method transfers constrained optimization problems into unconstrained problems. This can be done by adding an additional term, called penalty function, which consists of a penalty parameter multiplied by a measure of violation of the constraints. The following function Φ(t) is defined by

$$\Phi(t) = \begin{cases} 0 & if \quad t < 0 \\ Kt^2 & if \quad t \geq 0 \end{cases} \tag{10}$$

where K is some positive constant. This function is called penalty function, because it "penalizes" any number t which is greater than zero.

Penalty contact is based on nonlinear or linear spring elements, which are generated between the slave and the master surface (figure 16). In the current node-to-segment contact method (Nodes) the forces from this contact spring are distributed between the slave node and the nodes of the opposite master face. These nodes then form the so called contact element. The master surface is triangulated to achieve faster identification of the master face opposite of a given slave node. The master face for each slave node is determined by finding the triangle, of which center of gravity is closest to the slave node. If the orthogonal projection of the slave node is contained within this triangle, a contact spring element is generated consisting of the slave node and the master face the triangle belongs to, provided the node penetrates the structure or the clearance does not exceed a defined margin.

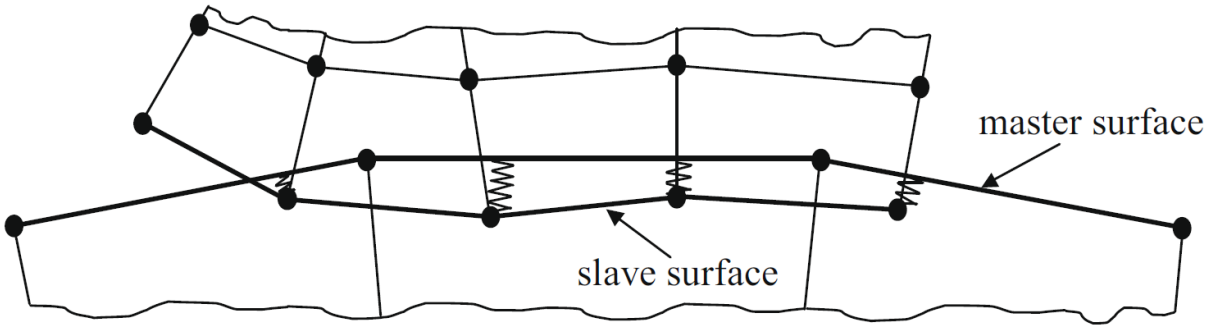


Figure 16. Illustration of the contact springs between the slave nodes and the master faces. (Zavarise & De Lorenzis, 2009)

As a default, this pairing of the slave nodes with the master faces is done every iteration up to the eighth iteration. In case user specifies the "SMALL SLIDING" parameter in the input file, the pairing is only done once per increment. Small sliding should be only used if the sliding is expected to be relatively small.

The behavior of the contact spring element in the Nodes version is either quasi bilinear, exponential or piecewise linear. The pressure exerted on the master face of a contact spring element with quasi bilinear behavior is given by

$$p = Kd\left[\frac{1}{2} + \frac{1}{\pi}tan^{-1}\frac{d}{\epsilon}\right], \tag{11}$$

where d is the overclosure and ϵ is defined as follows:

$$\epsilon = \sigma_\infty \frac{\pi}{K} \tag{12}$$

σ_∞ is the value for tension at large clearances. The pressure-overclosure -curve for the values of K = 10^3 and ϵ = 10^{-2} is given in the figure 17.

Large values of the spring constant K lead to hard contact. The values for K should be from 5 to 50 times the elastic modulus to achieve good results. Additionally, the value of the tension for negative overclosure must be given by the user. In case of a quasi bilinear contact spring element, the maximum clearance for the contact elements to be generated is $c_0\sqrt{spring\ area}$ if the spring area is positive and 10^{-10} otherwise. The value for c_0 may be defined by the user, but the default value is 10^{-3} (Dhondt, 2013).

If the exponential behavior is chosen for the contact spring elements, it is given by

$$p = p_0 \exp(\beta d), \tag{13}$$

where p_0 is the pressure at zero clearance and β is a coefficient without immediate physical significance. However, the user is expected to specify a value for c_0 which equals the clearance for which the pressure is 1% of p_0. β is then calculated as follows

$$\beta = \frac{\ln 100}{c_0} \tag{14}$$

A small value of c_0 leads to hard contact which also leads to slower convergence. (Dhondt, 2013) The pressure-overclosure –relationship for p_0 = 1 and c_0 = 0.5 is shown in the figure 18.

In case the user has experimental data for the pressure-overclosure relationship, a piecewise linear function can be defined by the user.

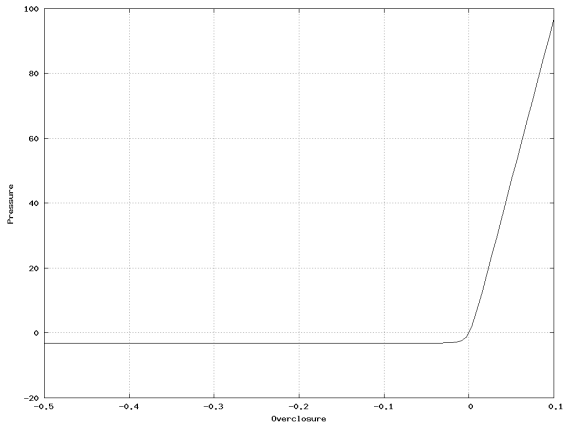


Figure 17. The quasi bilinear pressure-overclosure relationship.

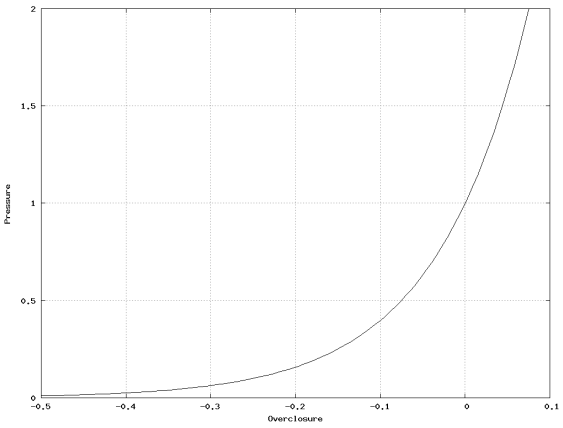


Figure 18. The exponential pressure-overclosure relationship.

After determining the contact pressure, the spring force can be calculated. The normal spring force is defined as the pressure multiplied by the spring area. If the frictional effects are taken into account, a tangential spring force is also present. It is defined as the shear stress multiplied by the spring area. The shear stress is a function of the relative displacement between the contact surfaces. The first part of the function is called a stick range, which is a linear function from zero to μ, where μ is the user specified friction coefficient. Λ is the tangent of the stick range and must also be provided by the user. The recommended value for Λ is approximately the spring constant divided by 100. The slip range is defined as a range where the shear stress is independent of the relative tangential displacement. The shear stress as a function of the relative tangential displacement is shown in the figure 19.

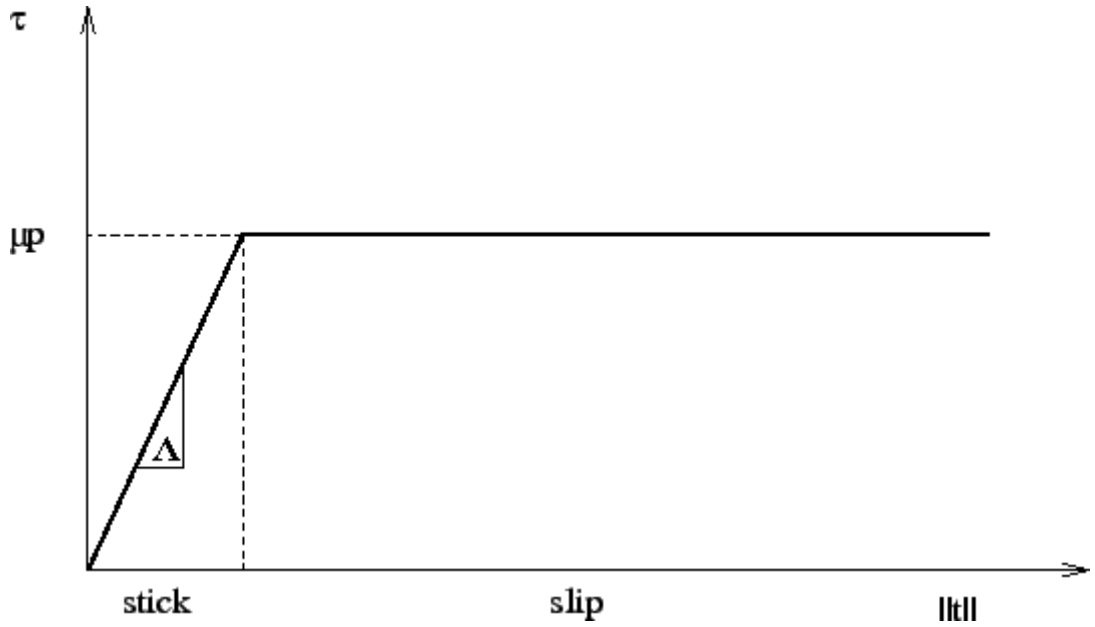


Figure 19. Shear stress as a function of the relative tangential displacement. (Dhondt, 2013)

3.3 Node-to-segment contact formulation

In the node-to-segment penalty contact definition, the pyramid shaped contact elements are created between each active slave node and the nodes of the opposite master face (figure 20). The general contact formulation is derived below.

ξ: local coordinate of the master surface
η: local coordinate of the master surface
p: location of the slave node
q: projection of **p** onto the master surface
$\mathbf{q}_j$: locations of the master nodes
m: normal vector of the master surface
n: unit vector of **m**
$\boldsymbol{\lambda}$: vector between **q** and **p**
u: displacement vector
f: contact force depending on clearance
$\mathbf{F}_p$: contact force vector at point **p**
$\mathbf{F}_j$: contact force vector at the master nodes
φ: shape function
δ_{ij}: Kronecker delta function
I: identity matrix

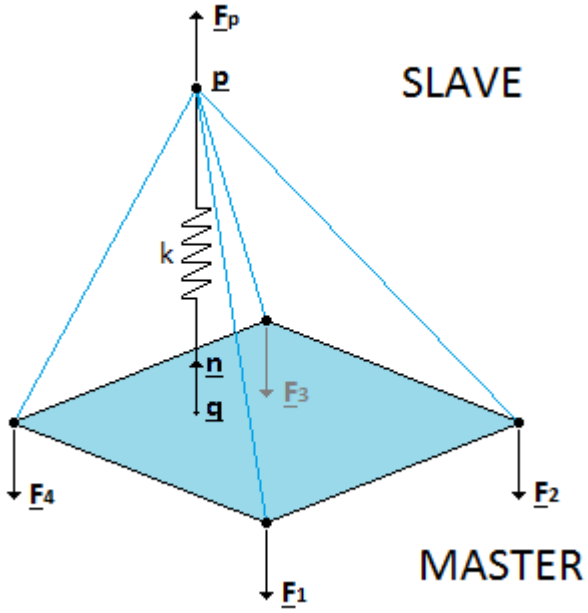


Figure 20. The principle of the node-to-segment contact formulation.

The normal vector of the master surface at the point **q**:

$$\boldsymbol{m} = \frac{\partial \boldsymbol{q}}{\partial \xi} \times \frac{\partial \boldsymbol{q}}{\partial \eta} \tag{15}$$

The unit vector of **m**:

$$\boldsymbol{n} = \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|} \tag{16}$$

The vector $\boldsymbol{\lambda}$ between the points **p** and **q**:

$$\boldsymbol{\lambda} = \boldsymbol{p} - \boldsymbol{q} \tag{17}$$

The scalar λ describes the clearance**:**

$$\lambda = \boldsymbol{\lambda} \cdot \boldsymbol{n} \tag{18}$$

The point **q** equals the sum over the multiplication of nodes and shape functions:

$$\boldsymbol{q} = \sum\nolimits_j \varphi_j(\xi,\eta)\boldsymbol{q}_j \tag{19}$$

The contact force f has been calculated from the area, for which the contact spring element is responsible, and the pressure-overclosure relationship. The contact force is transferred into a vector form ($\boldsymbol{F}_p$) (eq. 20).

$$\boldsymbol{F}_p = -f(\lambda)\boldsymbol{n} = -f(\boldsymbol{\lambda}\cdot\boldsymbol{n})\boldsymbol{n} \tag{20}$$

The derivatives of the clearance λ with respect to displacements $\mathbf{u}_i$ are calculated (eqs. 21a-21c), because they are needed later for the calculation of the stiffness matrix contributions.

$$\frac{\partial\lambda}{\partial\boldsymbol{u}_i} = \frac{\partial}{\partial\boldsymbol{u}_i}\left[\frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\cdot\left(\boldsymbol{p} - \sum\nolimits_j \varphi_j\boldsymbol{q}_j\right)\right] \tag{21a}$$

$$\frac{\partial\lambda}{\partial\boldsymbol{u}_i} = \frac{1}{\|\boldsymbol{m}\|}\left(\boldsymbol{p} - \sum\nolimits_j \varphi_j\boldsymbol{q}_j\right)\cdot\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}_i} - \frac{\boldsymbol{m}\cdot\left(\boldsymbol{p} - \sum_j \varphi_j\boldsymbol{q}_j\right)}{\|\boldsymbol{m}\|^2}\cdot\frac{\partial\|\boldsymbol{m}\|}{\partial\boldsymbol{u}_i} + \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|} \cdot\frac{\partial}{\partial\boldsymbol{u}_i}\left(\boldsymbol{p} - \sum\nolimits_j \varphi_j\boldsymbol{q}_j\right) \tag{21b}$$

$$\frac{\partial\lambda}{\partial\boldsymbol{u}_i} = \frac{\boldsymbol{\lambda}}{\|\boldsymbol{m}\|}\cdot\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}_i} - \frac{\lambda}{\|\boldsymbol{m}\|}\frac{\partial\|\boldsymbol{m}\|}{\partial\boldsymbol{u}_i} + \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\cdot\frac{\partial\boldsymbol{\lambda}}{\partial\boldsymbol{u}_i} \tag{21c}$$

The derivatives of the force vector $\boldsymbol{F}_p$ with respect to displacements $\mathbf{u}_i$ are calculated (eqs. 22a-22d) to determine the stiffness matrix.

$$\frac{\partial\boldsymbol{F}_p}{\partial\boldsymbol{u}_i} = -\frac{\partial f}{\partial\lambda}\left(\boldsymbol{n}\otimes\frac{\partial\lambda}{\partial\boldsymbol{u}_i}\right) - f\frac{\partial\boldsymbol{n}}{\partial\boldsymbol{u}_i} \tag{22a}$$

$$\frac{\partial\boldsymbol{F}_p}{\partial\boldsymbol{u}_i} = -\frac{\partial f}{\partial\lambda}\left(\frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\otimes\frac{\partial\lambda}{\partial\boldsymbol{u}_i}\right) - f\frac{\partial}{\partial\boldsymbol{u}_i}\left(\frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\right) \tag{22b}$$

$$\frac{\partial\boldsymbol{F}_p}{\partial\boldsymbol{u}_i} = -\frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\otimes\frac{\partial f}{\partial\lambda}\left\{\frac{\partial}{\partial\boldsymbol{u}_i}\left[\frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\cdot\left(\boldsymbol{p} - \sum\nolimits_j \varphi_j\boldsymbol{q}_j\right)\right]\right\} - \frac{f}{\|\boldsymbol{m}\|}\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}_i} + \frac{f}{\|\boldsymbol{m}\|^2}\boldsymbol{m} \otimes\frac{\partial\|\boldsymbol{m}\|}{\partial\boldsymbol{u}_i} \tag{22c}$$

When the relationship from equation (21c) is taken account, one ends up with the following form (eq. 22d):

$$\frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_i} = -\left(\frac{\partial f}{\partial \lambda}\frac{1}{\|\boldsymbol{m}\|^2}\right)\boldsymbol{m}\otimes\left(\lambda\cdot\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}_i}+\boldsymbol{m}\cdot\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{u}_i}-\lambda\frac{\partial\|\boldsymbol{m}\|}{\partial \boldsymbol{u}_i}\right) + \frac{f}{\|\boldsymbol{m}\|}\left(\boldsymbol{n}\otimes\frac{\partial\|\boldsymbol{m}\|}{\partial \boldsymbol{u}_i}-\frac{\partial \boldsymbol{m}}{\partial \boldsymbol{u}_i}\right) \tag{22d}$$

To proceed from this point, the derivatives of the normal vector **m** and its length $\|\boldsymbol{m}\|$as well as the vector $\boldsymbol{\lambda}$ with respect to displacements $\mathbf{u}_i$ must be calculated. The following (eqs. 23a-23b) is known from the equations (15) and (19).

$$\boldsymbol{m} = \frac{\partial \boldsymbol{q}}{\partial \xi}\times\frac{\partial \boldsymbol{q}}{\partial \eta}, \quad \boldsymbol{q} = \sum\nolimits_j \varphi_j(\xi,\eta)\boldsymbol{q}_j \tag{23a}$$

$$\boldsymbol{m} = \sum\nolimits_i\sum\nolimits_j \frac{\partial \varphi_i}{\partial \xi}\frac{\partial \varphi_j}{\partial \eta}\left(\boldsymbol{q}_i\times\boldsymbol{q}_j\right) \tag{23b}$$

The derivatives of the normal vector **m** with respect to displacements $\mathbf{u}_k$ (eqs. 24a-24b):

$$\begin{aligned}\frac{\boldsymbol{m}}{\boldsymbol{u}_k} = &\sum\nolimits_i\sum\nolimits_j\left[\frac{\partial^2\varphi_i}{\partial\xi^2}\frac{\partial\varphi_j}{\partial\eta}+\frac{\partial\varphi_i}{\partial\xi}\frac{\partial^2\varphi_j}{\partial\xi\partial\eta}\right]\left(\boldsymbol{q}_i\times\boldsymbol{q}_j\right)\otimes\frac{\partial\xi}{\partial\boldsymbol{u}_k}\\ &+\sum\nolimits_i\sum\nolimits_j\left[\frac{\partial\varphi_i}{\partial\xi}\frac{\partial^2\varphi_j}{\partial\eta^2}+\frac{\partial^2\varphi_i}{\partial\xi\partial\eta}\frac{\partial\varphi_j}{\partial\eta}\right]\left(\boldsymbol{q}_i\times\boldsymbol{q}_j\right)\otimes\frac{\partial\eta}{\partial\boldsymbol{u}_k}\\ &+\sum\nolimits_i\sum\nolimits_j\frac{\partial\varphi_i}{\partial\xi}\frac{\partial\varphi_j}{\partial\eta}\left(\delta_{ik}\boldsymbol{I}\times\boldsymbol{q}_j-\delta_{jk}\boldsymbol{I}\times\boldsymbol{q}_i\right)\end{aligned} \tag{24a}$$

$$\begin{aligned}\frac{\boldsymbol{m}}{\boldsymbol{u}_k} = &\left[\frac{\partial^2\boldsymbol{q}}{\partial\xi^2}\times\frac{\partial\boldsymbol{q}}{\partial\eta}+\frac{\partial\boldsymbol{q}}{\partial\xi}\times\frac{\partial^2\boldsymbol{q}}{\partial\xi\partial\eta}\right]\otimes\frac{\partial\xi}{\partial\boldsymbol{u}_k}+\left[\frac{\partial\boldsymbol{q}}{\partial\xi}\times\frac{\partial^2\boldsymbol{q}}{\partial\eta^2}+\frac{\partial^2\boldsymbol{q}}{\partial\xi\partial\eta}\times\frac{\partial\boldsymbol{q}}{\partial\eta}\right]\otimes\frac{\partial\eta}{\partial\boldsymbol{u}_k}\\ &+\sum\nolimits_i\sum\nolimits_j\left(\frac{\partial\varphi_i}{\partial\xi}\frac{\partial\varphi_j}{\partial\eta}-\frac{\partial\varphi_j}{\partial\xi}\frac{\partial\varphi_i}{\partial\eta}\right)\left(\boldsymbol{I}\times\boldsymbol{q}_j\right)\delta_{ik}\end{aligned} \tag{24b}$$

The derivatives of $\|\boldsymbol{m}\|$ with respect to displacements $\mathbf{u}_k$ (eq. 25):

$$\frac{\partial\|\boldsymbol{m}\|}{\partial\boldsymbol{u}_k} = \frac{\boldsymbol{m}}{\|\boldsymbol{m}\|}\cdot\frac{\partial\boldsymbol{m}}{\partial\boldsymbol{u}_k} \tag{25}$$

The derivatives of the vector $\boldsymbol{\lambda}$ with respect to displacements $\mathbf{u}_i$ (eqs. 26a-26d):

$$\frac{\partial\boldsymbol{\lambda}}{\partial\boldsymbol{u}_i} = \frac{\partial}{\partial\boldsymbol{u}_i}\left(\boldsymbol{p}-\sum\nolimits_j\varphi_j(\xi,\eta)\boldsymbol{q}_j\right) \tag{26a}$$

$$\frac{\partial\boldsymbol{\lambda}}{\partial\boldsymbol{u}_i} = \frac{\partial\boldsymbol{p}}{\partial\boldsymbol{u}_i}-\sum\nolimits_j\left[\boldsymbol{q}_j\otimes\left(\frac{\partial\varphi_j}{\partial\xi}\frac{\partial\xi}{\partial\boldsymbol{u}_i}+\frac{\partial\varphi_j}{\partial\eta}\frac{\partial\eta}{\partial\boldsymbol{u}_i}\right)+\varphi_j\frac{\partial\boldsymbol{q}_j}{\partial\boldsymbol{u}_i}\right] \tag{26b}$$

$$\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{u}_i} = \boldsymbol{I}\delta_{ip} - \sum_j \left[\boldsymbol{q}_j \otimes \left(\frac{\partial \varphi_j}{\partial \xi}\frac{\partial \xi}{\partial \boldsymbol{u}_i} + \frac{\partial \varphi_j}{\partial \eta}\frac{\partial \eta}{\partial \boldsymbol{u}_i} \right) + \boldsymbol{I}\varphi_j \delta_{ij} \right] \tag{26c}$$

$$\frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{u}_i} = \boldsymbol{I}\delta_{ip} - \sum_j \left[\frac{\partial \boldsymbol{q}}{\partial \xi} \otimes \frac{\partial \xi}{\partial \boldsymbol{u}_i} + \frac{\partial \boldsymbol{q}}{\partial \eta} \otimes \frac{\partial \eta}{\partial \boldsymbol{u}_i} + \varphi_i \big(1 - \delta_{ip}\big)\boldsymbol{I} \right] \tag{26d}$$

The orthogonality condition states that, the inner product of two orthogonal vectors equals zero. The orthogonality between vector $\boldsymbol{\lambda}$ and the derivative of the point $\mathbf{q}$ with respect to ξ-direction yields the following equations (eqs. 27a-28d):

$$\boldsymbol{\lambda} \perp \frac{\partial \boldsymbol{q}}{\partial \xi} \Rightarrow \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} = 0 \tag{27a}$$

$$\left(\boldsymbol{p} - \sum_i \varphi_i(\xi, \eta)\boldsymbol{q}_i \right) \cdot \left(\sum_j \frac{\partial \varphi_j}{\partial \xi}\boldsymbol{q}_j \right) = 0 \tag{27b}$$

Differentiation of the equation (26) yields to a following:

$$\begin{aligned} &\left[d\boldsymbol{p} - \sum_i \left(\frac{\partial \varphi_i}{\partial \xi}\boldsymbol{q}_i d\xi + \frac{\partial \varphi_i}{\partial \eta}\boldsymbol{q}_i d\eta + \varphi_i d\boldsymbol{q}_i \right) \right] \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} + \boldsymbol{\lambda} \\ &\qquad \cdot \left[\sum_j \left(\frac{\partial^2 \varphi_j}{\partial \xi^2}\boldsymbol{q}_j d\xi + \frac{\partial^2 \varphi_j}{\partial \xi \partial \eta}\boldsymbol{q}_j d\eta + \frac{\partial \varphi_j}{\partial \xi} d\boldsymbol{q}_j \right) \right] = 0 \end{aligned} \tag{28a}$$

$$\begin{aligned} &\left[d\boldsymbol{p} - \frac{\partial \boldsymbol{q}}{\partial \xi} d\xi - \frac{\partial \boldsymbol{q}}{\partial \eta} d\eta - \sum_i \varphi_i d\boldsymbol{q}_i \right] \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} + \boldsymbol{\lambda} \\ &\qquad \cdot \left[\frac{\partial^2 \boldsymbol{q}}{\partial \xi^2} d\xi + \frac{\partial^2 \boldsymbol{q}}{\partial \xi \partial \eta} d\eta + \sum_j \frac{\partial \varphi_j}{\partial \xi} d\boldsymbol{q}_j \right] = 0 \end{aligned} \tag{28b}$$

Rearranging the terms of the equation (28b) produces the following relationship:

$$\begin{aligned} &\left(-\frac{\partial \boldsymbol{q}}{\partial \xi} \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} + \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{q}}{\partial \xi^2} \right) d\xi + \left(-\frac{\partial \boldsymbol{q}}{\partial \eta} \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} + \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{q}}{\partial \xi \partial \eta} \right) d\eta \\ &\qquad = -\left(d\boldsymbol{p} - \sum_i \varphi_i d\boldsymbol{q}_i \right) \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} - \boldsymbol{\lambda} \cdot \left(\sum_j \frac{\partial \varphi_j}{\partial \xi} d\boldsymbol{q}_j \right) \end{aligned} \tag{28c}$$

or

$$\begin{aligned} &\left(-\frac{\partial \boldsymbol{q}}{\partial \xi} \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} + \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{q}}{\partial \xi^2} \right) d\xi + \left(-\frac{\partial \boldsymbol{q}}{\partial \eta} \cdot \frac{\partial \boldsymbol{q}}{\partial \xi} + \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{q}}{\partial \xi \partial \eta} \right) d\eta \\ &\qquad = -\frac{\partial \boldsymbol{q}}{\partial \xi} \cdot d\boldsymbol{p} + \sum_i \left[\left(\varphi_i \frac{\partial \boldsymbol{q}}{\partial \xi} - \frac{\partial \varphi_i}{\partial \xi}\boldsymbol{\lambda} \right) \cdot d\boldsymbol{q}_i \right] \end{aligned} \tag{28d}$$

The orthogonality between vector $\boldsymbol{\lambda}$ and the derivative of the point $\mathbf{q}$ with respect to η-direction is calculated similarly (eqs. 29-30).

$$\boldsymbol{\lambda} \perp \frac{\partial \boldsymbol{q}}{\partial \eta} \Rightarrow \boldsymbol{\lambda} \cdot \frac{\partial \boldsymbol{q}}{\partial \eta} = 0 \tag{29}$$

The final form of the equation becomes (see eqs. 27a-28d for comparison):

$$\begin{aligned} \left(-\frac{\partial \boldsymbol{q}}{\partial \xi} \cdot \frac{\partial \boldsymbol{q}}{\partial \eta} + \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{q}}{\partial \xi \partial \eta}\right) d\xi + \left(-\frac{\partial \boldsymbol{q}}{\partial \eta} \cdot \frac{\partial \boldsymbol{q}}{\partial \eta} + \boldsymbol{\lambda} \cdot \frac{\partial^2 \boldsymbol{q}}{\partial \eta^2}\right) d\eta \\ = -\frac{\partial \boldsymbol{q}}{\partial \eta} \cdot d\boldsymbol{p} + \sum\nolimits_i \left[\left(\varphi_i \frac{\partial \boldsymbol{q}}{\partial \eta} - \frac{\partial \varphi_i}{\partial \eta} \boldsymbol{\lambda}\right) \cdot d\boldsymbol{q}_i\right] \end{aligned} \tag{30}$$

At this point, there are two equations (eqs. 28d,30) and two unknown variables, $d\xi$ and $d\eta$. After solving the 2x2 equation system, one ends up with the following relationships (eqs. 31-32).

$$\frac{\partial \xi}{\partial \boldsymbol{q}_i} = \frac{\partial \xi}{\partial \boldsymbol{u}_i}, \quad \frac{\partial \xi}{\partial \boldsymbol{p}} = \frac{\partial \xi}{\partial \boldsymbol{u}_p} \tag{31}$$

$$\frac{\partial \eta}{\partial \boldsymbol{q}_i} = \frac{\partial \eta}{\partial \boldsymbol{u}_i}, \quad \frac{\partial \eta}{\partial \boldsymbol{p}} = \frac{\partial \eta}{\partial \boldsymbol{u}_p} \tag{32}$$

The force vector at the master nodes is calculated by the shape functions (eq. 33).

$$\boldsymbol{F}_j = -\varphi_j(\xi, \eta) \boldsymbol{F}_p \tag{33}$$

The stiffness matrix contributions (eq. 34) can be calculated using the results (eqs. 31-32) from the equations (28d) and (30).

$$\frac{\partial \boldsymbol{F}_j}{\partial \boldsymbol{u}_i} = -\boldsymbol{F}_p \otimes \left(\frac{\partial \varphi_j}{\partial \xi} \frac{\partial \xi}{\partial \boldsymbol{u}_i} + \frac{\partial \varphi_j}{\partial \eta} \frac{\partial \eta}{\partial \boldsymbol{u}_i}\right) - \varphi_j \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_i} \tag{34}$$

3.3.1 Issues

The main issues encountered with the node-to-segment contact formulation are caused by the relative tangential displacement of the contact surfaces. The measure of the penetration is calculated by the closest point projection from the slave node onto the master surface. The normal force is then defined to be coaxial with the normal of the master face at the corresponding location. The non-smooth surface causes discontinuities in the contact force on the slave node when it slides from one master face to another. This is illustrated in the figure 21. Another problem may occur when individual slave nodes slide off the boundaries of the master surface. Both aforementioned scenarios cause non-physical jumps in the contact forces and lead to convergence difficulties. The proposed segment-to-segment approach deals with both of the issues and is expected to lead to better convergence and stability (Puso & Laursen, 2004).

The node-to-segment formulation in CalculiX is so called single pass algorithm, which means that only the nodes of the slave surface are prevented from penetrating the opposite master surface. Due to this property a significant loss of accuracy is encountered if the mesh of the slave surface is much coarser than the mesh of the master surface. Moreover, the node-to-segment formulations do not pass the patch test. The segment-to-segment formulations are also expected to overcome these drawbacks. (Puso & Laursen, 2004)

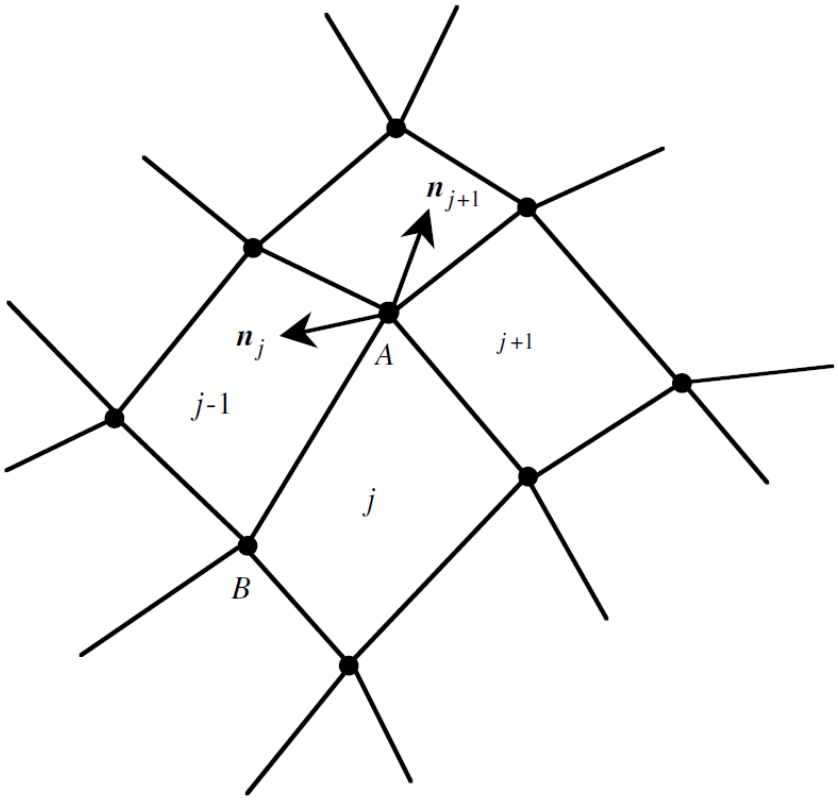


Figure 21. The normals for the faces j and j+1 are shown. The discontinuity in contact forces occurs when the slave node slides from one face to another. (Puso & Laursen, 2004)

4 New formulation

This chapter presents the proposed segment-to-segment formulation and two different approaches to obtain the discretization schemes. The main difference between these approaches is that the first one results in a biased treatment of the contact surfaces, whereas the second one does not. Both types of discretization schemes were implemented in CalculiX for comparison. Furthermore, the linear approach to the contact calculation is introduced. The problems encountered by the new formulation are gone through and the solutions are proposed. The discretization scheme resulting in a biased treatment of the contact surfaces was not implemented by the author and the cutting algorithm for the second discretization technique already existed, but everything else presented in this chapter has been programmed by the author.

4.1 Segment-to-segment contact formulation

In the node-to-segment formulation, contact springs are formed at the locations of the active slave nodes. This actually means that the magnitude of the force experienced by the slave node equals the force of the contact spring. However, as the other side of the contact spring is located at some random point within the master face, the forces at the master nodes are obtained by the shape functions. For segment-to-segment formulations, neither of the sides of the contact spring is located at a node. This means that for a segment-to-segment formulation the contact forces must also be transferred to

the slave nodes from the different spring locations determined by the applied discretization scheme.

In the segment-to-segment contact definition, the contact elements are created between the slave nodes within one face and the nodes of the opposite master face. These contact elements consist of certain amount of contact springs created between the slave and the master surface (figure 22). The formulation applies regardless of the amount of integration points (possible contact spring locations) within one face. The modifications required to the original node-to-segment version are derived below.

ξ_m: local coordinate of the master surface
η_m: local coordinate of the master surface
ξ_s: local coordinate of the slave surface
η_s: local coordinate of the slave surface
p: location of the integration point
q: projection of **p** to the master surface
$\mathbf{q}_j$: locations of the master nodes
m: normal vector of the master surface
n: unit vector of **m**
$\boldsymbol{\lambda}$: vector between **q** and **p**
u: displacement vector
f: contact force depending on clearance
$\mathbf{F}_p$: contact force vector at point **p**
$\mathbf{F}_{jm}$: contact force vector at the master nodes
$\mathbf{F}_{js}$: contact force vector at the slave nodes
φ: shape function
δ_{ij}: Kronecker delta function
I: identity matrix

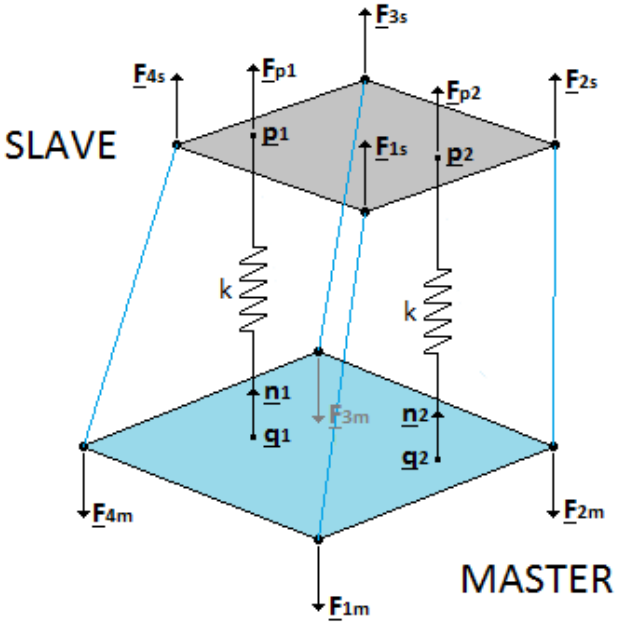


Figure 22. The principle of the segment-to-segment contact formulation.

The contact force at the master nodes is still calculated as in the section 3.3 (eq. 35).

$$\boldsymbol{F}_{j_m} = -\varphi_{j_m}(\xi_m, \eta_m)\boldsymbol{F}_p \tag{35}$$

The contact force must now be calculated also for the slave nodes. The nodal values are calculated similarly compared to the master nodes (eq. 36).

$$\boldsymbol{F}_{j_s} = \varphi_{j_s}(\xi_s, \eta_s)\boldsymbol{F}_p \tag{36}$$

The following relationship is needed for the calculation of the new stiffness matrix contributions.

$$\boldsymbol{u}_p = \sum_i \varphi_{i_s}\, \boldsymbol{u}_{i_s} \tag{37}$$

The first component of the stiffness matrix contributions equals the one calculated in the section 3.3.

$$\frac{\partial \boldsymbol{F}_{j_m}}{\partial \boldsymbol{u}_{i_m}} = -\boldsymbol{F}_p \otimes \left(\frac{\partial \varphi_{j_m}}{\partial \xi_m} \frac{\partial \xi_m}{\partial \boldsymbol{u}_{i_m}} + \frac{\partial \varphi_{j_m}}{\partial \eta_m} \frac{\partial \eta_m}{\partial \boldsymbol{u}_{i_m}} \right) - \varphi_{j_m} \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_{i_m}} \tag{38}$$

The following three components are derived due to the segment-to-segment definition (eqs. 39-41).

$$\frac{\partial \boldsymbol{F}_{j_s}}{\partial \boldsymbol{u}_{i_s}} = \varphi_{j_s} \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_{i_s}} = \varphi_{j_s} \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_p} \cdot \frac{\partial \boldsymbol{u}_p}{\partial \boldsymbol{u}_{i_s}} = \varphi_{i_s} \varphi_{j_s} \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_p} \tag{39}$$

$$\frac{\partial \boldsymbol{F}_{j_m}}{\partial \boldsymbol{u}_{i_s}} = \frac{\partial \boldsymbol{F}_{j_m}}{\partial \boldsymbol{u}_p} \cdot \frac{\partial \boldsymbol{u}_p}{\partial \boldsymbol{u}_{i_s}} = \varphi_{i_s} \frac{\partial \boldsymbol{F}_{j_m}}{\partial \boldsymbol{u}_p} \tag{40}$$

$$\frac{\partial \boldsymbol{F}_{j_s}}{\partial \boldsymbol{u}_{i_m}} = \varphi_{j_s} \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_{i_m}} \tag{41}$$

If the amount of nodes within one master face equals k and the amount of nodes within one slave face equals l, one ends up with the following form of the stiffness matrix (eq. 42). In this case, the dimensions of the stiffness matrix would be $3(k+l) \times 3(k+l)$.

$$[\boldsymbol{S}] = \begin{bmatrix} \frac{\partial F_{x_{1_m}}}{\partial u_{x_{1_m}}} & \cdots & \frac{\partial F_{x_{1_m}}}{\partial u_{z_{k_m}}} & \frac{\partial F_{x_{1_m}}}{\partial u_{x_{1_s}}} & \cdots & \frac{\partial F_{x_{1_m}}}{\partial u_{z_{l_s}}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{z_{k_m}}}{\partial u_{x_{1_m}}} & \cdots & \frac{\partial F_{z_{k_m}}}{\partial u_{z_{k_m}}} & \frac{\partial F_{z_{k_m}}}{\partial u_{x_{1_s}}} & \cdots & \frac{\partial F_{z_{k_m}}}{\partial u_{z_{l_s}}} \\ \frac{\partial F_{x_{1_s}}}{\partial u_{x_{1_m}}} & \cdots & \frac{\partial F_{x_{1_s}}}{\partial u_{z_{k_m}}} & \frac{\partial F_{x_{1_s}}}{\partial u_{x_{1_s}}} & \cdots & \frac{\partial F_{x_{1_s}}}{\partial u_{z_{l_s}}} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_{z_{l_s}}}{\partial u_{x_{1_m}}} & \cdots & \frac{\partial F_{z_{l_s}}}{\partial u_{z_{k_m}}} & \frac{\partial F_{z_{l_s}}}{\partial u_{x_{1_s}}} & \cdots & \frac{\partial F_{z_{l_s}}}{\partial u_{z_{l_s}}} \end{bmatrix} \tag{42}$$

4.2 Gauss discretization scheme

The Gauss discretization scheme defines contact between the slave and the master surface by the integration points placed in the certain constant locations determined by the Gaussian quadrature. Generally, the faces within the contact surface are quadrilateral or triangular. For the quadrilateral face, there are one, four or nine integration points depending on the element type. For triangular faces, one or three point scheme is used. The Gauss formulation results in a biased treatment of the contact surfaces because the integration point scheme is based only on the slave surface mesh. The coordinates for the integration points lying on the slave faces are given as follows:

Quadrilateral face:

1-point scheme:	9-point scheme:
$p_1(\xi_1,\eta_1),\ \ \xi_1=0,\ \ \eta_1=0,$	$p_1(\xi_1,\eta_1),\ \ \xi_1=-\frac{\sqrt{3}}{\sqrt{5}},\ \ \eta_1=-\frac{\sqrt{3}}{\sqrt{5}},$
4-point scheme:	$p_2(\xi_2,\eta_2),\ \ \xi_2=0,\ \ \eta_2=-\frac{\sqrt{3}}{\sqrt{5}},$
$p_1(\xi_1,\eta_1),\ \ \xi_1=-\frac{\sqrt{3}}{3},\ \ \eta_1=-\frac{\sqrt{3}}{3},$	$p_3(\xi_3,\eta_3),\ \ \xi_3=\frac{\sqrt{3}}{\sqrt{5}},\ \ \eta_3=-\frac{\sqrt{3}}{\sqrt{5}}$
$p_2(\xi_2,\eta_2),\ \ \xi_2=\frac{\sqrt{3}}{3},\ \ \eta_2=-\frac{\sqrt{3}}{3},$	$p_4(\xi_4,\eta_4),\ \ \xi_4=-\frac{\sqrt{3}}{\sqrt{5}},\ \ \eta_4=0,$
$p_3(\xi_3,\eta_3),\ \ \xi_3=-\frac{\sqrt{3}}{3},\ \ \eta_3=\frac{\sqrt{3},}{3}$	$p_5(\xi_5,\eta_5),\ \ \xi_5=0,\ \ \eta_5=0,$
$p_4(\xi_4,\eta_4),\ \ \xi_4=\frac{\sqrt{3}}{3},\ \ \eta_4=\frac{\sqrt{3},}{3}$	$p_6(\xi_6,\eta_6),\ \ \xi_6=\frac{\sqrt{3}}{\sqrt{5}},\ \ \eta_6=0,$
	$p_7(\xi_7,\eta_7),\ \ \xi_7=-\frac{\sqrt{3}}{\sqrt{5}},\ \ \eta_7=\frac{\sqrt{3}}{\sqrt{5}}$
	$p_8(\xi_8,\eta_8),\ \ \xi_8=0,\ \ \eta_8=\frac{\sqrt{3}}{\sqrt{5}},$
	$p_9(\xi_9,\eta_9),\ \ \xi_9=\frac{\sqrt{3}}{\sqrt{5}},\ \ \eta_9=\frac{\sqrt{3}}{\sqrt{5}},$

where -1$\leq \xi,\eta \leq 1$

Triangular face:

1-point scheme:	3-point scheme:
$p_1(\xi_1,\eta_1),\ \ \xi_1=\frac{1}{3},\ \ \eta_1=\frac{1}{3},$	$p_1(\xi_1,\eta_1),\ \ \xi_1=\frac{1}{6},\ \ \eta_1=\frac{1}{6},$
	$p_2(\xi_2,\eta_2),\ \ \xi_2=\frac{2}{3},\ \ \eta_2=\frac{1}{6},$
	$p_3(\xi_3,\eta_3),\ \ \xi_3=\frac{1}{6},\ \ \eta_3=\frac{2,}{3}$

where 0 $\leq \xi,\eta \leq 1$

4.3 Gauss Cut discretization scheme

In the Gauss Cut discretization scheme, the integration points are determined by the triangulation of the contact area. Within each triangle, a constant amount of integration points is created according to the Gaussian quadrature. The experiments were conducted with one-, three, seven-, and twelve-point schemes. The locations of the integration points within the face of the element are not fixed, because the triangulation depends on the relative displacement of the contact surfaces. The Gauss Cut formulation results in an unbiased treatment of contact surfaces because the integration point scheme is based on the cutting of the slave and the master surface. The triangulation and integration points for the seven-point scheme within one quadrilateral master face are shown below (figure 23).

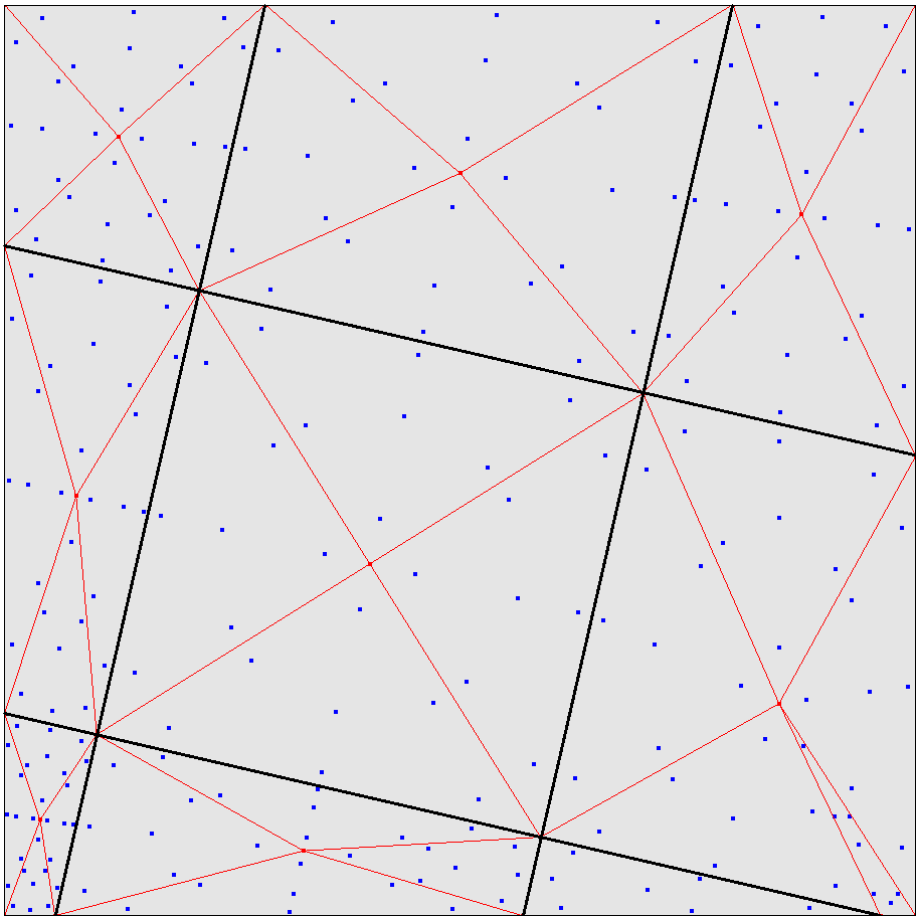


Figure 23. The triangulation and the integration points within one master face for the seven-point Gauss Cut discretization scheme. The black straight lines are the borderlines of the opposite slave faces. The integration points (blue dots) are determined by the triangulation (red lines).

4.3.1 Friction

The segment-to-segment contact formulation requires the slave faces to be divided to a certain amount of integration points. Contact variables such as pressure and overlapping as well as the internal state variables needed for frictional calculations are stored at the integration points. However, in the Gauss Cut contact definition new integration points are created at the beginning of each increment due to the displacements of the contact surfaces. After this is done, the old integration points are not valid anymore and the internal state variables must be transferred to the new integration points. The problem was solved by interpolating the values to the new integration points at the beginning of the new increment. At first, the interpolation was performed just by extrapolating the nodal values at the end of the increment and then interpolating these nodal values back to the new integration points within the face. However, it was noticed that this method was too inaccurate.

The new method is based on the Delanay triangulation of each face. The vertices of the triangles are determined by the integration points. Additionally, the corner nodes are added as extra triangulation points, such that the whole face is covered by triangles. Each pair of neighboring corner nodes must belong to one common triangle so that the edges of the face do not become concave. The values for the extra triangulation points (corner nodes) are determined by the extrapolation (section 4.3.3). The values for the nodes belonging to more than one slave face are determined by taking the mean value of the results provided by the all common faces. This way there will be no jumps in the values between adjacent faces.

The Delanay triangulation forms a unique triangulation from a set of points within a plane. This triangulation maximizes the minimum angle of all angles of the triangles and thus avoids skinny triangles. The fast algorithm used to form the Delanay triangulation was written by S. W. Sloan (1987). The process is started by selecting three points to form a “supertriangle” which completely covers the whole slave face. Initially, the triangulation consists of a single triangle defined by the supertriangle vertices. Then, the points to be triangulated are introduced one at the time. When a new point is introduced into the triangulation, the first thing is to find the triangle which encloses this point. After this, three new triangles are formed by connecting the new point to the vertices of the triangle, in which the new point is located. This procedure is continued until the last point is introduced into the triangulation. When the triangulation is completed, the final triangulation is obtained by removing all of the triangles which contain supertriangle vertices.

In general, the size and the shape of the supertriangle can be chosen arbitrarily, as long as it contains all of the points to be triangulated. However, if the vertices of the supertriangle are not far enough, the triangulation might not form a convex hull of the corresponding set of the points, i.e. the boundary of the triangulation may become locally concave. This cannot be accepted, because if the whole face is not covered by

the triangulation there exists possibility that some of the new integration points lie outside the triangulated area. This can be overcome by moving the vertices of the supertriangle further away. The coordinates of the vertices must be large enough to make sure that the triangle which would be formed with one supertriangle vertex, one node, and one integration point, always contains smaller angle than the triangle formed with two nodes and the integration point closest to the border of the element face. This is illustrated in the figure 24. The angle α formed with the blue line must always be greater than the angle β formed with the red line.

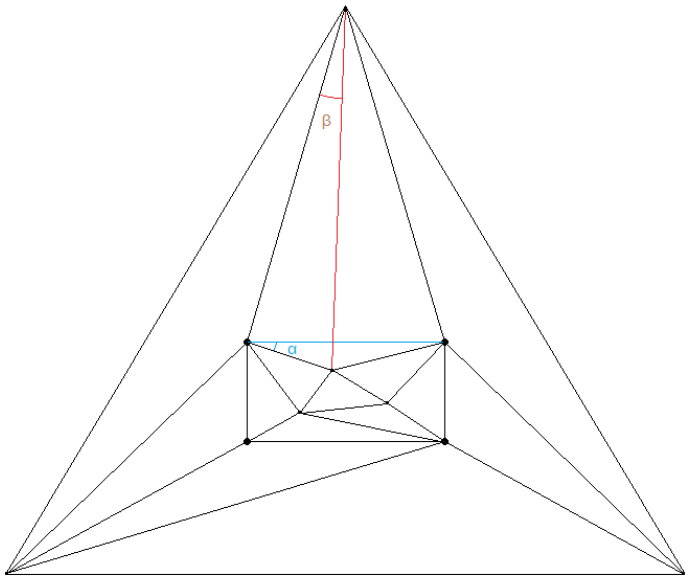


Figure 24. The Delanay triangulation for a set of points including the triangles formed with the supertriangle. The quadrilateral formed by the black dots (nodes) represents an element face, which has three integration points. The three coordinates furthest away from the middle are the vertices of the supertriangle.

Based on the experiments conducted with many turbine engine part models, the aforementioned condition is satisfied when the vertices of the supertriangle are located at least 10^5 times further away from the center of gravity of the face compared to any triangulated point. This is enough if there are no points closer to the border than approximately 0,1‰ of the distance between the border of the face and the center of the face. However, even if the concave border of the face now occurs and some of the new integration points are created outside the triangulated area, the values can be accurately approximated from the closest triangle, because the uncovered area is very narrow.

Another similar problem found was that the new integration point did not seem to lie within any triangle, even though the triangulation formed a convex hull. Actually, the point was found to be in-between the triangles. This is caused by numerical inaccuracy of the floating point numbers. However, in this case the interpolation from either of

these triangles is possible due to the fact that the new integration point is also located very close to these triangles.

After the triangulation is completed, the correct triangle is identified in two steps. The first step is to identify the triangle, of which center of gravity is closest to the new integration point. However, this does not guarantee that the integration point lies within the corresponding triangle. In the second step, the cross products between vectors $\overline{AB}$ and $\overline{Ap}$ as well as $\overline{AB}$ and $\overline{AC}$ are calculated (figure 25). If the inner product of these cross products is negative, point p lies outside triangle ABC, if positive, it may lie inside the triangle. This is expressed in the following equations:

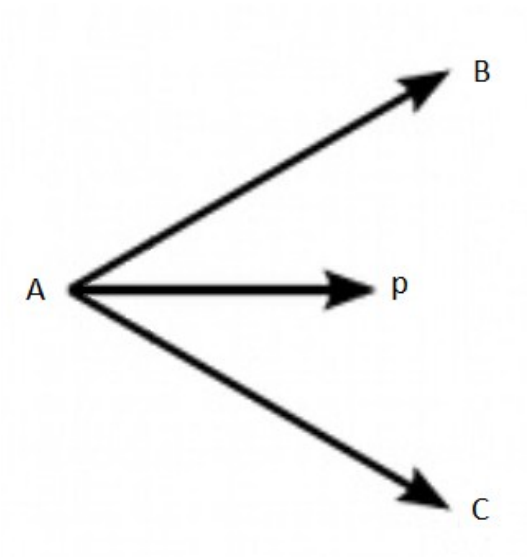


Figure 25. Vectors $\overline{AB}$, $\overline{Ap}$ and $\overline{AC}$ constructed from the triangle ABC.

$$(\overline{AB} \times \overline{Ap}) \cdot (\overline{AB} \times \overline{AC}) \geq 0 \Rightarrow \textit{point p possibly inside the triangle} \quad (43)$$

$$(\overline{AB} \times \overline{Ap}) \cdot (\overline{AB} \times \overline{AC}) < 0 \Rightarrow \textit{point p outside the triangle} \quad (44)$$

If the aforementioned procedure indicates that the integration point is not necessarily located outside the triangle, the location relative to the vectors $\overline{BC}$ and $\overline{CA}$ must also be checked. If the point lies outside the triangle relative to the edge $\overline{AB}$, the same procedure is conducted to the neighboring triangle containing this edge. The information about the neighboring triangles is stored by the triangulation algorithm and is available. This procedure is continued as long as the correct triangle is found, after which the plane equation (section 4.3.3) is applied to the corresponding triangle to find the interpolated values.

4.3.2 Visualization

The postprocessor CalculiX GraphiX is used for the visualization of the results-file produced by the solver. This requires the visualized values to be stored at the nodes. However, contact variables are initially stored at the integration points as written in the previous section. The problem was solved by the same extrapolation method which was used to extrapolate the nodal values of the internal state variables needed for friction calculation.

However, the visualization posed a problem at the boundary areas of the contact. If only the part of the face is in contact, not all of the integration points have formed a contact element. Furthermore, if there is only few contact elements in the one corner of the

element face, the interpolated nodal values may not be accurate at all. This caused high negative peak values at the boundary areas, but also rare positive peak values.

Solution was found by introducing two new conditions. At first, if the node lies outside the border of the master face, the extrapolation is not conducted for that node. The second condition states that the nodal values may only get values between the lowest and the highest value of the integration points within the face. Due to the fact that there are quite many integration points per face, this should not cause significant error.

4.3.3 Extrapolation

The same extrapolation code was used for both of the aforementioned problems, but the extrapolation method depends on the amount of integration points. The possible amount of integration points may vary from one to more than a hundred. From the mathematical point of view this has an affect for the chosen extrapolation method. If the amount of integration points is below the number of nodes within the face, the corresponding system of equations becomes underdetermined. In the opposite scenario, the system of equations becomes overdetermined. In the case of quadratic elements, the values of the middle nodes are just determined by taking the mean value of the neighboring corner nodes. Therefore there is maximum number of four nodal values to extrapolate. The following different methods are used:

Table 2. Extrapolation methods.

	Quadrilateral face:	**Triangular face:**
$\boldsymbol{i < 3}$	Constant extrapolation	Constant extrapolation
$\boldsymbol{i = 3}$	Plane equation	Shape functions
$\boldsymbol{i = 4}$	Shape functions	Shape functions (least squares method)
$\boldsymbol{i > 4}$	Shape functions (least squares method)	Shape functions (least squares method)

i represents the number of integration points.

Constant extrapolation

In the first scenario, if $i < 3$, a constant extrapolation is performed. The mean value of the integration points is just simply transferred to all nodes. In the case of one integration point, the value of the integration point is just copied to all nodes.

Plane equation

In the second scenario, if the element face is quadrilateral and $i = 3$, a plane equation is created from the values of three known points within the face. The plane equation is of the form

$$ax + by + cz + d = 0, \tag{45}$$

of which coefficients can be solved from the following equations:

$$\begin{aligned} ax_1 + by_1 + cz_1 + d &= 0, \\ ax_2 + by_2 + cz_2 + d &= 0, \\ ax_3 + by_3 + cz_3 + d &= 0. \end{aligned} \tag{46}$$

The coefficients are determined in the following equations (47-50):

$$a = \frac{-d}{D}\begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} = \frac{-d}{D}(y_2z_3 + y_1z_2 + z_1y_3 - z_2y_3 - y_1z_3 - z_1y_2) \tag{47}$$

$$b = \frac{-d}{D}\begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} = \frac{-d}{D}(x_1z_3 + z_2x_3 + z_1x_2 - x_1z_2 - x_2z_3 - z_1x_3) \tag{48}$$

$$c = \frac{-d}{D}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = \frac{-d}{D}(x_1y_2 + y_1x_3 + x_2y_3 - x_1y_3 - y_1x_2 - y_2x_3) \tag{49}$$

$$D = \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix} = x_1y_2z_3 + y_1z_2x_3 + z_1x_2y_3 - x_1z_2y_3 - y_1x_2z_3 - z_1y_2x_3 \tag{50}$$

If z is used to describe extrapolated value, the equation for z becomes

$$z = -\frac{ax + by + d}{c}, \tag{51}$$

where x and y are the two-dimensional coordinates of the corresponding face and d is any nonzero number.

Shape functions

In the third scenario, if the amount of integration points equals the amount of corner nodes, the system of equations is solved just by inverting the equation (52). The equation states that

$$x_j = \sum_i \varphi_i(\xi_j, \eta_j) x_i, \tag{52}$$

where j denotes integration points and i nodes. This equation expressed in the matrix form is written below.

$$\{x\}_{int\ points} = [\boldsymbol{A}]\{x\}_{nodes} \tag{53}$$

The above equation is solved by calculating the inverse of the matrix [**A**]. The nodal values are then found as follows

$$\{x\}_{nodes} = [\boldsymbol{A}]^{-1}\{x\}_{int\ points} \tag{54}$$

Shape functions by applying the least squares method

In the last scenario, if there are more integration points than nodes, the shape functions are still used to obtain the nodal values, but the method of least squares is applied. The least squares method is used to give the best-fitting curve to a given set of points by minimizing the squares of the offsets. Instead of using the absolute values of the offsets, the sum over the squares of the offsets is used. Now, the resulting equation becomes continuously differentiable, which is a prerequisite for finding the minimum of the sum of the squares. (Miller, 2006)

$\hat{x}_j$: actual value at the integration point
x_j: predicted value at the integration point
x_i: predicted value at the node
$\varphi(\xi_j, \eta_j)$: shape function

$$x_j = \sum_i \varphi_i(\xi_j, \eta_j) x_i \tag{55}$$

The least squares method is applied as follows:

$$f = \sum_j \left(\hat{x}_j - x_j\right)^2 = \sum_j \left[\hat{x}_j - \sum_i \varphi_i(\xi_j, \eta_j) x_i\right]^2 \tag{56}$$

The sum of the squares (eq. 56) of the residuals must be minimized in order to get the extrapolated nodal values. The minimum of the sum of the squares is found by setting the gradient to zero (eq. 57).

$$\frac{\partial f}{\partial x_k} = \sum_j 2\left[\hat{x}_j - \sum_i \varphi_i(\xi_j,\eta_j)x_i\right]\left[-\varphi_k(\xi_j,\eta_j)\right] = 0 \tag{57a}$$

$$\sum_i \left[\sum_j \varphi_i(\xi_j,\eta_j)\varphi_k(\xi_j,\eta_j)\right] x_i = \sum_j \hat{x}_j\,\varphi_k(\xi_j,\eta_j) \tag{57b}$$

This is a system of linear equations of the form $\mathbf{Ax}=\mathbf{b}$ of which solution can be expressed in the form $\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}$.

4.4 Linear approach

The so called linear version was developed for both, the Gauss and the Gauss Cut discretization scheme. In the linear version, c_0 (the distance, which determines whether the contact element is created) is set to zero and the quasi bilinear pressure-overclosure -relationship is made a truly linear function (figure 26). Also the following variables are kept constant during the iteration loop: spring area, the point of projection of the integration point onto the master surface, the normal vector of the master surface at the aforementioned point (equations 16 and 19). As a result, equations needed to calculate the stiffness matrix become much simpler. The purpose of this linear version is to achieve better convergence characteristics. The simplification of the stiffness contributions are shown in the following equations (eqs. 58-60).

$$\frac{\partial \xi}{\partial \boldsymbol{u}_i} = \frac{\partial \eta}{\partial \boldsymbol{u}_i} = \frac{\partial \boldsymbol{n}}{\partial \boldsymbol{u}_i} = 0 \tag{58}$$

$$\frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_i} = -\frac{\partial f}{\partial \boldsymbol{\lambda}}\left(\boldsymbol{n} \otimes \frac{\partial \lambda}{\partial \boldsymbol{u}_i}\right) - f\frac{\partial \boldsymbol{n}}{\partial \boldsymbol{u}_i} = -\frac{\partial f}{\partial \boldsymbol{\lambda}}\left(\boldsymbol{n} \otimes \frac{\partial \lambda}{\partial \boldsymbol{u}_i}\right) \tag{59}$$

$$\frac{\partial \boldsymbol{F}_j}{\partial \boldsymbol{u}_i} = -\boldsymbol{F}_p \otimes \left(\frac{\partial \varphi_j}{\partial \xi}\frac{\partial \xi}{\partial \boldsymbol{u}_i} + \frac{\partial \varphi_j}{\partial \eta}\frac{\partial \eta}{\partial \boldsymbol{u}_i}\right) - \varphi_j \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_i} = -\varphi_j \frac{\partial \boldsymbol{F}_p}{\partial \boldsymbol{u}_i} \tag{60}$$

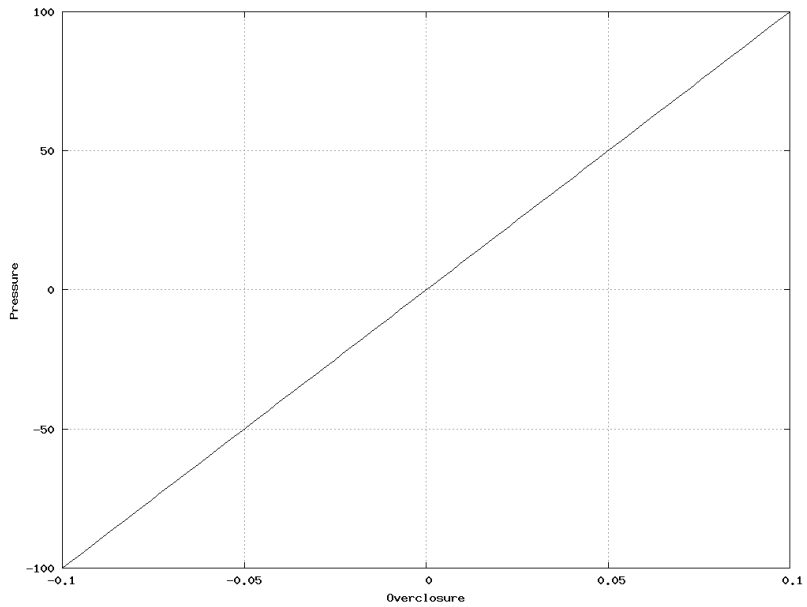


Figure 26.The truly linear pressure-overclosure relationship.

4.4.1 Tensional forces

Even though c_0 is set to zero, there exists a possibility that contact elements cause tensional forces. This problem occurs if the prerequisite for the existence of a contact element is not checked often enough. For nonlinear versions, it is checked for each iteration up to eighth iteration, and only once per increment if small sliding is activated. For the linear version, prerequisites for contact elements are checked every iteration without upper limit to avoid unrealistic tensional contact forces. Additionally, this means that small sliding cannot be used with linear version.

4.4.2 Convergence and incrementation control

For the current node-to-segment penalty method, the convergence criteria are satisfied provided that there is no change in the number of contact elements between two consecutive iterations and the residual force as well as the displacements are small enough. For the segment-to-segment implementations the change in the number of contact elements is allowed if it lies within one per mille. This condition was introduced, because of the huge increase in the amount of contact elements. It does not have a significant effect on results, but it allows the solution to converge usually a few iterations earlier for the big real models consisting of hundreds of thousands of contact elements.

The automatic incrementation control used to control the size of the upcoming increment is based on the number of iterations which was required to reach the convergence in the previous increment. This turned out to cause problems with linear segment-to-segment formulations. For the current node-to-segment penalty method, the increment size is increased if two subsequent increments are completed for less than five iterations. Similarly, the increment is decreased if the previous increment required more than ten iterations. The aforementioned numbers are default values and can be changed by the user.

However, it seems that the convergence behavior of the linear segment-to-segment formulations significantly differs from the previous node-to-segment formulation. It was noticed that often increments require more than ten iterations and decreasing the increment size would not necessarily reduce the required amount of iterations. This is partly caused by the fact that the number of contact elements is not frozen after the eighth iteration. Based on the experiments, it seems that the best results were obtained just by never decreasing the increment size based on the number of iterations required by the previous increment. Similarly, it seems that the requirements for the next increment to be increased could be loosened.

4.4.3 Procedure to detect divergence

The fact that the number of contact elements is not frozen after the eighth iteration in the linear version causes additional problems with the procedure to detect divergence. The procedure to detect divergence is important since it has a significant effect on the

calculation times and the convergence rate of examples. If the example would have converged in the following iterations, but the program stops iterating because the solution seems to diverge, the divergence is probably determined too early. This leads to not converged examples and longer computational times due to the unnecessarily small increment sizes. If the opposite occurs, the program keeps iterating even though it seems highly probable that the solution is really diverging. This also leads to longer computational times. The goal is to come up with most efficient criteria for detection of divergence so that all examples would reach convergence as quickly as possible.

The criteria used in the current node-to-segment penalty version to detect divergence are quite simple. First of all, divergence cannot be detected if the number of contact elements is changing from one iteration to another. However, the number of contact elements is frozen after eighth iteration, so the detection of divergence is always possible after this point. It is also possible before the eighth iteration if the number of contact elements is already constant between consecutive iterations. Then, if the requirement for the number of contact elements is satisfied, divergence is detected in case the absolute values of the residual forces of the following two iterations are bigger than the current residual force. Additionally, the residual force must not be so small that it would allow convergence.

However, the method presented does not give efficient results for the linear version, because now the number of contact elements might never stabilize. Therefore, one must also take into account the change in the number of contact elements as well as the residual forces. In the other words, the change in the residual forces must be allowed to have more “room”, if the amount of contact elements is not constant. This also requires the default maximum number of iterations to be increased, because the stabilization of the contact elements might take now much longer than eight iterations. The criteria developed, mainly by studying the convergence characteristics of many examples, are presented in the figure 27. Not too much time was used to optimize these rules, so it is likely that better rules can be found by experimentation.

The first condition (I) states that if the residual forces of the current and the previous iteration are bigger than the minimum mean residual force of the two consecutive iterations, then divergence is possible. The second condition (II) states that, if the contact elements are stabilized, divergence is possible. This is actually the same requirement that was already introduced for the node-to-segment method. Finally the last condition (III) states that, if the number of contact elements is not stabilizing, and subsequent residual forces form a repetitive pattern, divergence is possible.

C = Number of contact elements
R = Absolute value of the residual force
*Residual force must not be small enough to allow convergence

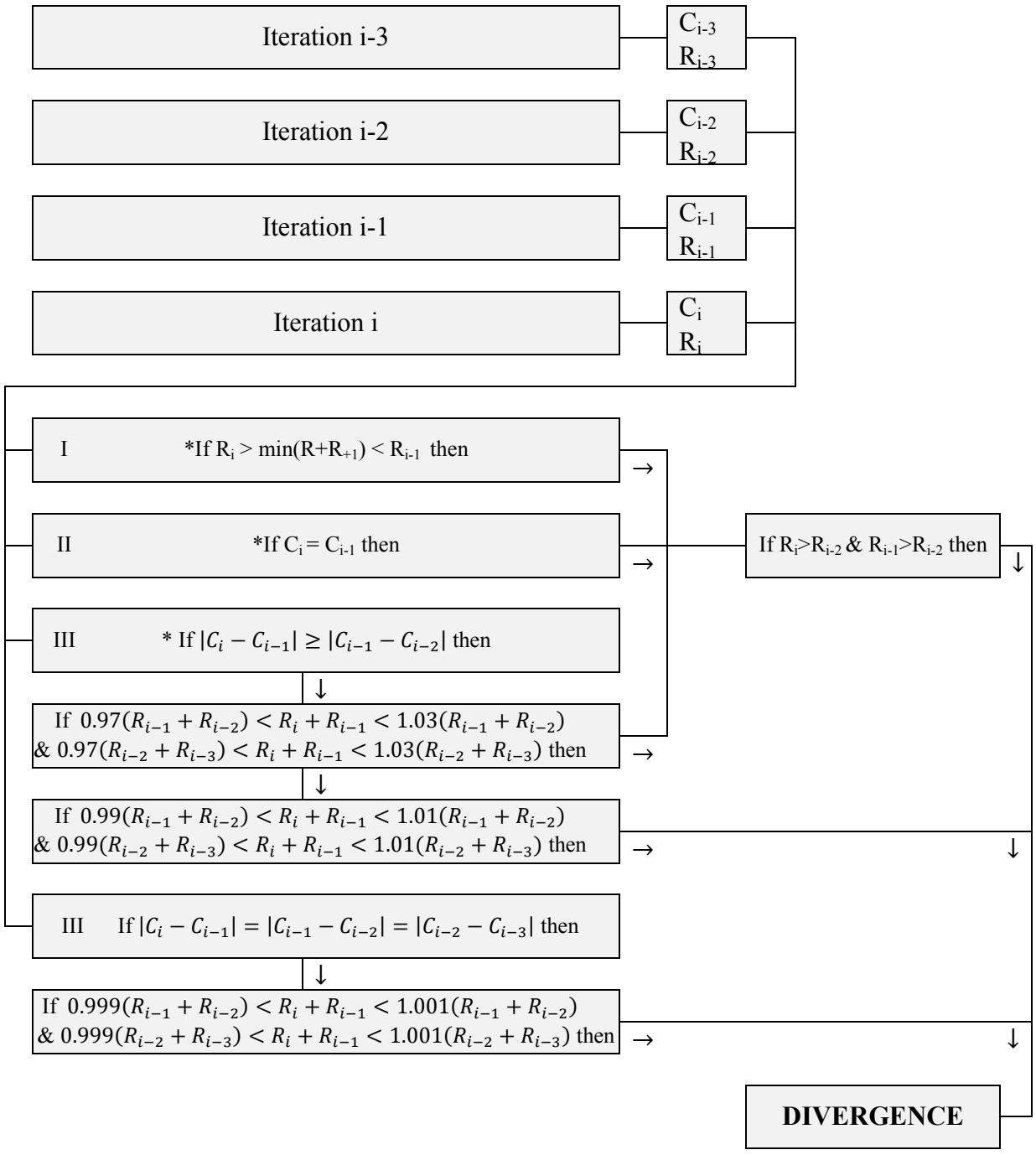


Figure 27. The criteria for the procedure to detect diverging residuals.

5 Test examples

The relatively simple test examples were the first ones used to test the new contact formulations. They were used during the programming to find possible problems, but also after the implementation of the contact formulation to compare the results and the stability of the different versions. These examples are small, easily modifiable and relatively fast to run.

In this section the comparison between the results of the test examples ran by different versions is conducted. The comparison between different CalculiX versions and Abaqus is done, because Abaqus is considered to produce reliable results. Due to the limited space, not all possible quantities are compared. The comparison is made between von Mises stresses, displacements, and contact variables. If plasticity is taken account, it is also presented.

The stability and efficiency of the contact formulation is compared by varying the coordinates, boundary conditions or loadings of the examples. Unit system used throughout this section is {mm, N, s, K}. If the units are not specified within the text, the aforementioned units may be assumed.

5.1 Cube

The first test example was the cube example. It consists of two cubes, which are placed one upon the other (figure 28). The lower cube is fixed in all directions (x, y and z) at the bottom surface. The upper cube is fixed in x- and y-directions at the top surface. The constant pressure loading is applied to the top surface of the upper cube and has magnitude of 100. The specifications for the cube example are shown in the following table.

Table 3. The specifications for the cube example.

Parameter	**Lower cube**	**Upper cube**
Dimensions (x,y,z)	1.0, 1.0, 1.0	0.5, 0.5, 0.5
Elastic modulus	$2.1 * 10^5$	$2.1 * 10^5$
Density	$7.8 * 10^{-9}$	$7.8 * 10^{-9}$
Mesh (elements)	16*16*16	16*16*16
Element type	C3D8	C3D8

Contact parameters

$K = 10^7$
$\sigma_\infty = 10/\pi$
$\mu = 0.2$
$\Lambda = 5*10^5$

The default setup of the cube example is shown in the figure 28. The location of the upper cube is defined by the coordinates given to the input file. The location in the figure 28 corresponds the coordinates (x,y,z) = (0,0,0) and rotation = 0.

The setup of the cube example, which is used in the next figures, is shown in the figure 29. The coordinates chosen for this setup are defined as follows: (x,y,z) = (0.3,0.4,0) and rotation = 77°. The vertical position of the upper cube is altered only in the figure to illustrate the contact elements. The contact elements are shown in blue color.

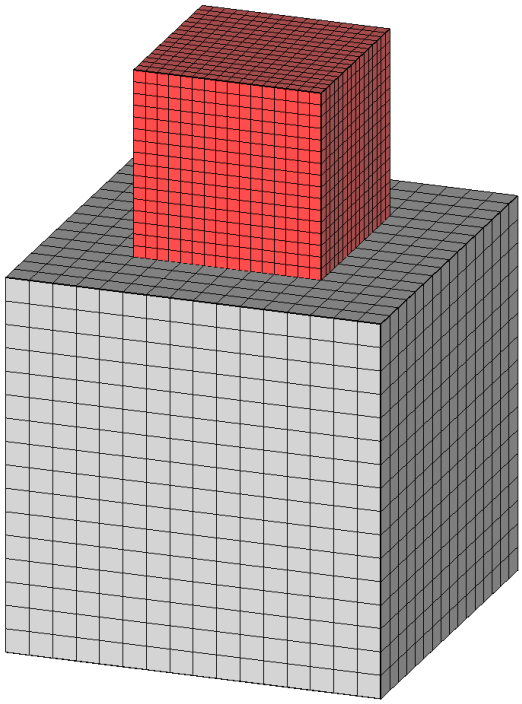


Figure 28. The default setup of the cube example.

5.1.1 Results

The results of the cube example were compared between Abaqus STS (segment-to-segment), Abaqus NTS (node-to-segment) and following penalty contact definitions implemented in CalculiX: Nodes (2.6.2), Linear Gauss, Gauss Cut (GC) and Linear Gauss Cut (LGC).

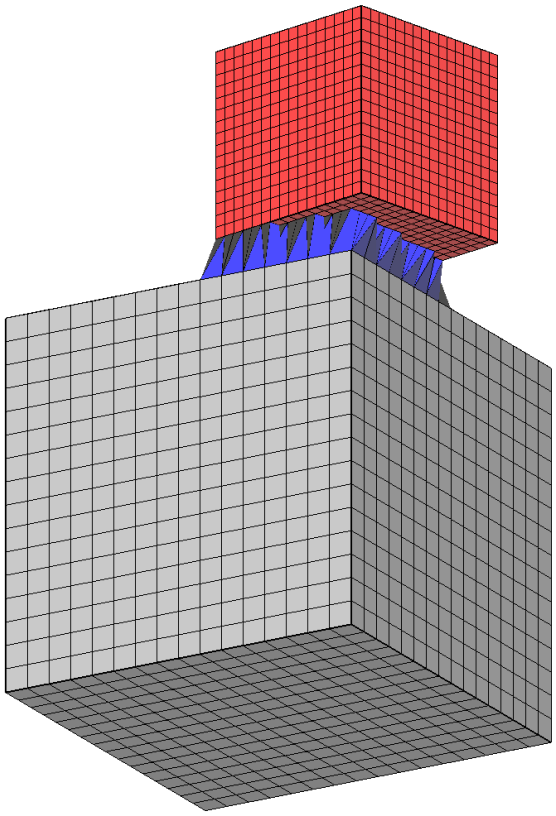


Figure 29. The setup of the cube example which is used for comparison of the results.

At first, the von Mises stress for the whole model is given in the figures 30a-30e. Minor differences can be found between the different versions. First of all, there is a clear difference between node-to-segment and segment-to-segment formulations. The stresses near the contact surfaces are much smoother for the segment-to-segment formulations and the peak values are lower. At the end, all CalculiX versions seem to give quite similar stress distribution. The maximum values from CalculiX are quite similar to Abaqus.

The displacements between different CalculiX versions seem to be really close to each other (figures 31a-31f). However, Abaqus gives slightly higher displacements.

The results from the contact surface are presented in the Appendix I. Von Mises stress at the slave surface is really similar for all segment-to-segment formulations. Additionally, the results from both node-to-segment versions are also similar, but the coarser stress distribution becomes very clear.

Furthermore, there are no significant differences in the contact variables either. The Gauss Cut versions give coarser results at the boundary of the master surface due to the contact variable extrapolation. In general, the segment-to-segment versions seem to give slightly higher peak values.

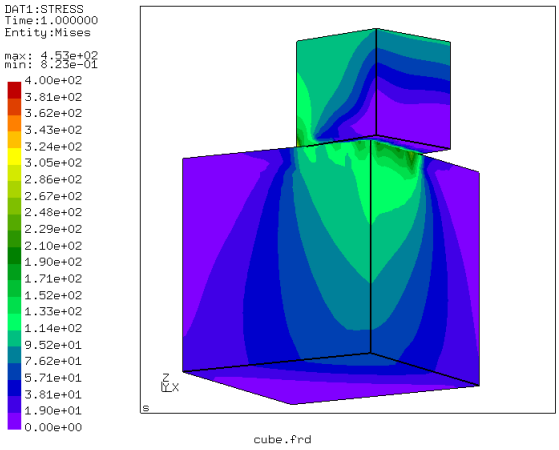


Figure 30a. Von Mises, Abaqus NTS.

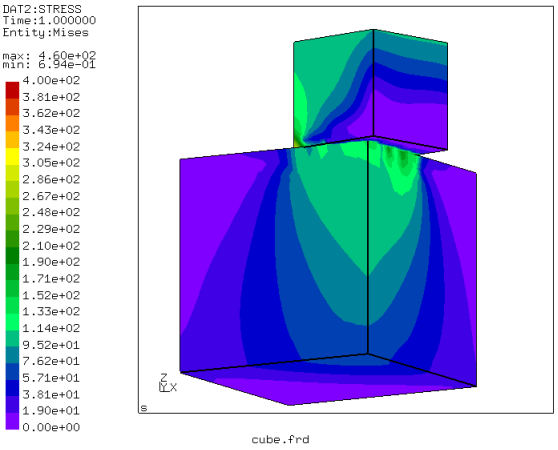


Figure 30b. Von Mises, Nodes.

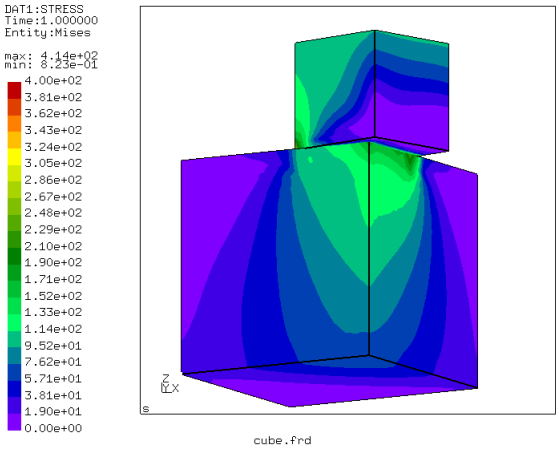


Figure 30c. Von Mises, Abaqus STS.

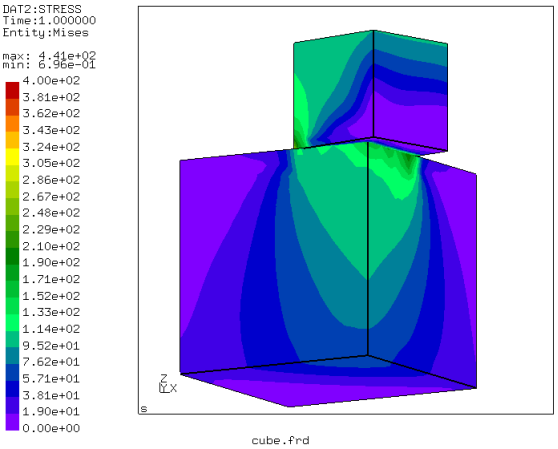


Figure 30d. Von Mises, Linear Gauss.

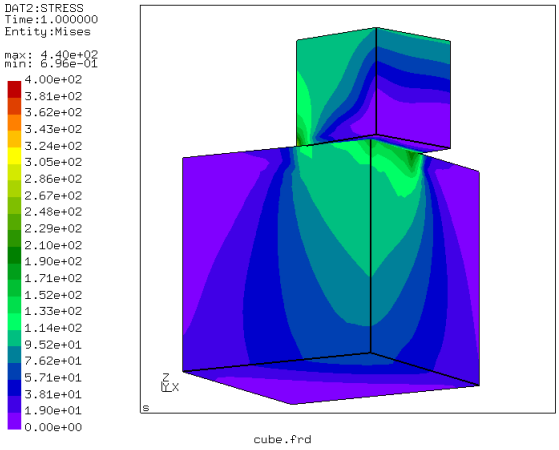


Figure 30e. Von Mises, GC7.

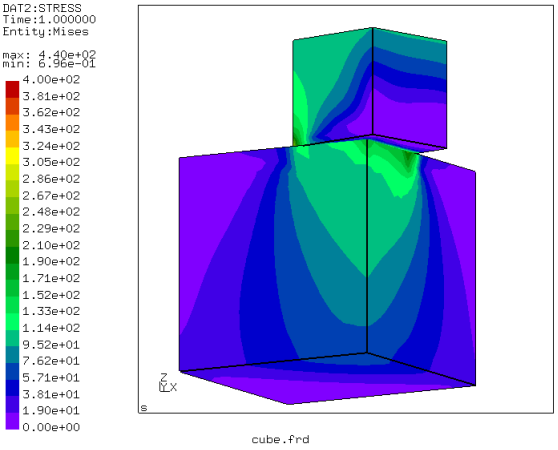
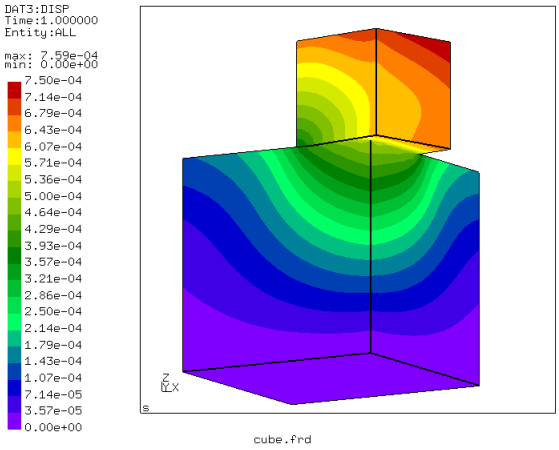


Figure 30f. Von Mises, LGC7.



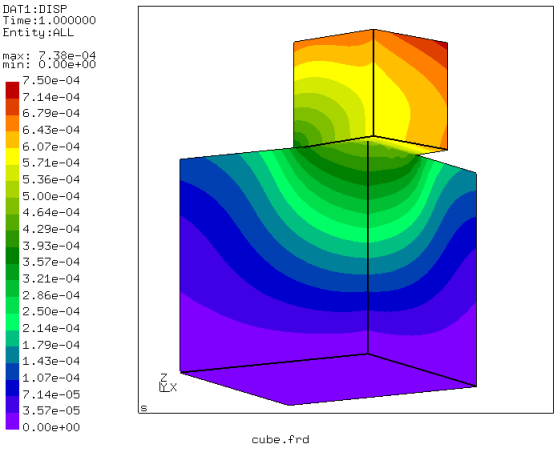
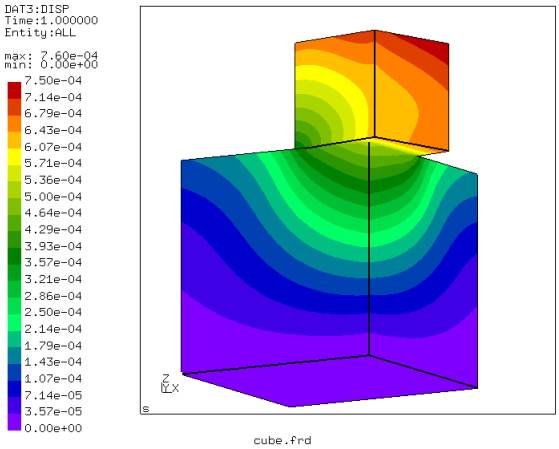


Figure 31a. Displacements, Abaqus NTS. **Figure 31b. Displacements, Nodes.**



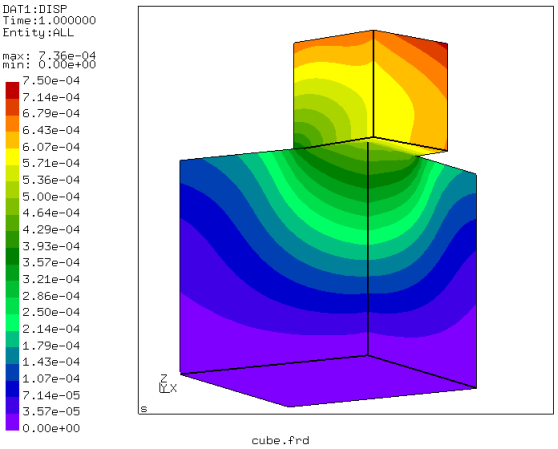
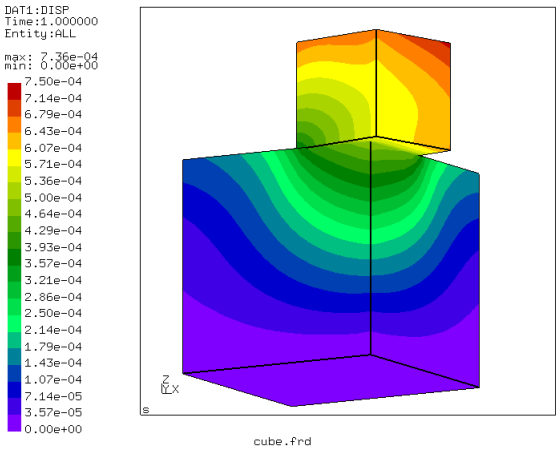


Figure 31c. Displacements, Abaqus STS. **Figure 31d. Displacements, Linear Gauss.**



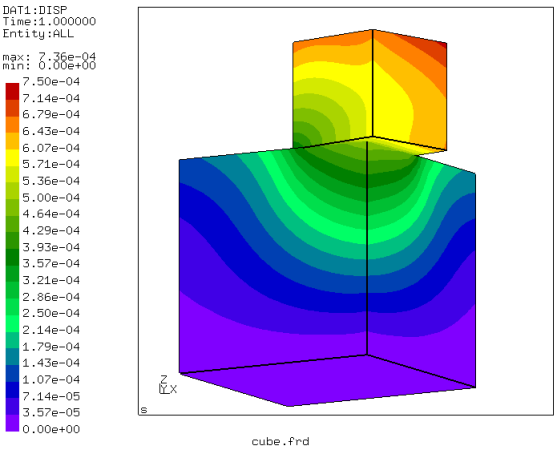


Figure 31e. Displacements, GC7. **Figure 31f. Displacements, LGC7.**

5.1.2 Stability and efficiency

The cube example was used to determine the initial estimate about the stability and the efficiency of the segment-to-segment penalty contact formulations. The cube example was run thousand times by varying the location and the rotational position of the upper cube. The important numbers from the tests are the number of iterations required and the number of runs which did not converge. The number of iterations required describes the efficiency and the number of not converged runs describes the instability of the contact definition. The comparison is made between eight different contact implementations with linear and quadratic contact surfaces. The results are presented below in the figure 32.

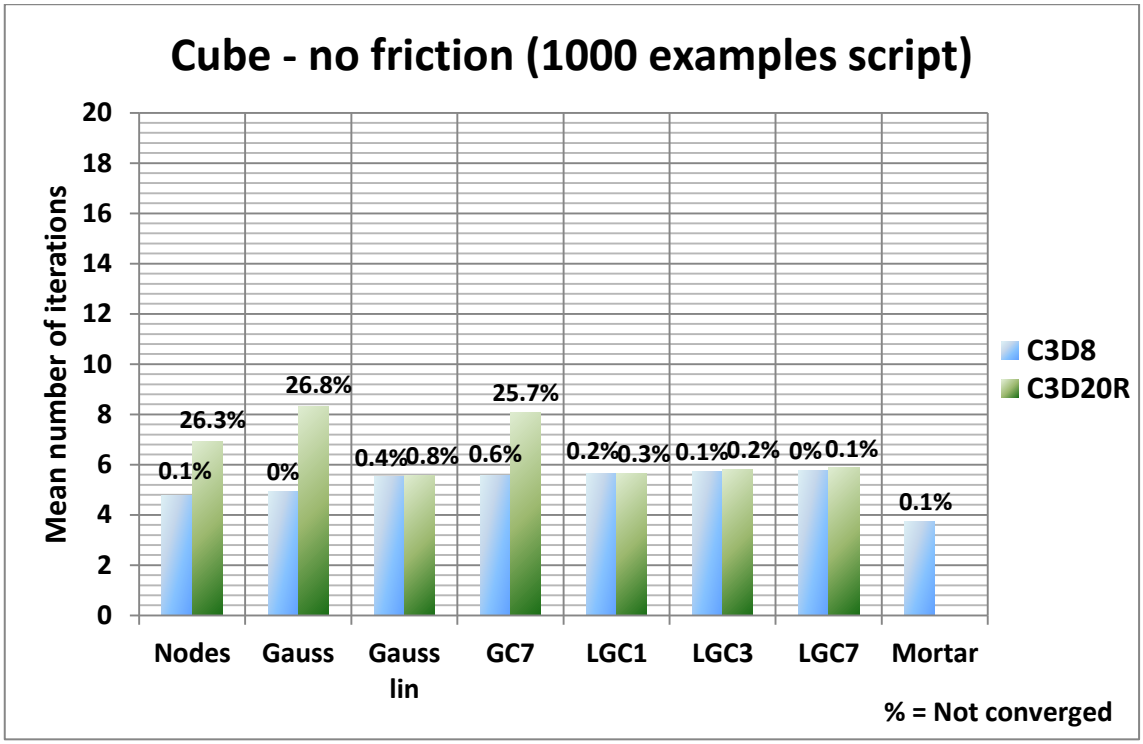


Figure 32. The results for the cube example.

The first thing to note is that all three nonlinear versions (Nodes, Gauss, GC7) have problems with quadratic contact surfaces. About one quarter of the different positions of the upper cube did not converge. With linear elements, all versions actually seem to perform really good without convergence problems.

Furthermore, the linear versions perform really good independent of the element type. No differences between the linear and the quadratic elements can be noticed. However, if Linear Gauss, LGC1, LGC3 and LGC7 are compared, there is slight indication of the fact that the more integration points, the better the convergence.

5.2 Punch

The punch example is similar to the first one. It still consists of two cubes placed one upon the other, but the bottom surface of the upper cube is now curved. The boundary conditions are similar compared to the cube example. Also the constant pressure loading applied to the top surface of the upper cube is similar with magnitude of 100. The specifications of the punch example are shown in the following table.

Table 4. The specifications for the punch example.

Parameter	Lower cube	Upper cube
Dimensions (x,y,z)	1.0, 1.0, 1.0	0.3826, 0.3826, 0.3826
Elastic modulus	$2.1 * 10^5$	$2.1 * 10^5$
Density	$7.8 * 10^{-9}$	$7.8 * 10^{-9}$
Mesh (elements)	16*16*16	16*16*16
Element type	C3D8	C3D8
Radius of curvature	-	0.5 (bottom surface)

Contact parameters

$K = 10^7$
$\sigma_\infty = 10/\pi$
$\mu = 0.2$
$\Lambda = 5*10^5$

The default setup of the punch example is shown in the figure 33. The location of the upper cube is defined by the coordinates given to the input file. The location in the figure 33 corresponds the coordinates (x,y,z) = (0,0,0) and rotation = 0.

The setup of the punch example, which was run for the next figures, is shown in the figure 34. The coordinates chosen for this setup are defined as follows: (x,y,z) = (0.5,0.3,-0.0002) and rotation = 66°. The slightly negative z-coordinate is used to ensure initial contact. The vertical position of the upper cube is altered only in the figure to illustrate the contact elements. The contact elements are shown in blue color.

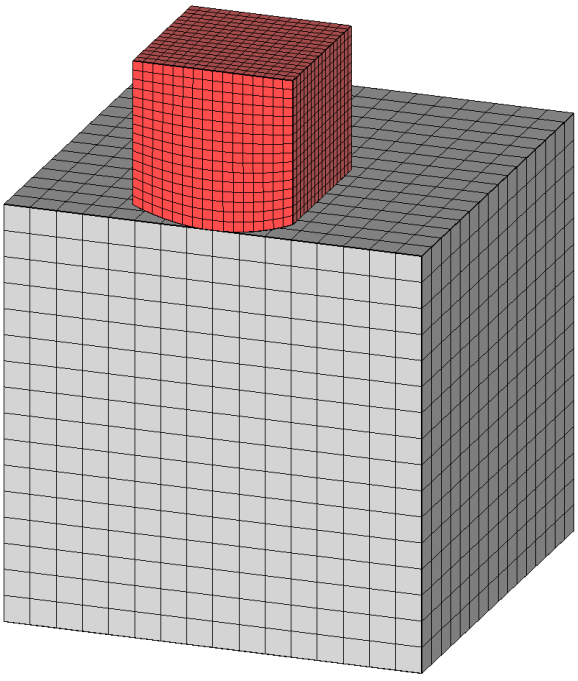


Figure 33. The default setup of the punch example.

5.2.1 Results

The results of the punch example were compared between Abaqus STS (segment-to-segment), Abaqus NTS (node-to-segment) and the following penalty contact definitions implemented in CalculiX: Nodes (2.6.2), Linear Gauss, Gauss Cut (GC) and Linear Gauss Cut (LGC).

At first, the von Mises stress for the whole model is given in the figures 35a-35f. The shape of the stress distribution is almost identical between all versions.

For displacements, differences between node-to-segment and segment-to-segment implementations can be found (figures 36a-36f). The segment-to-segment versions give about 10% bigger displacements. The linear versions are also giving slightly higher displacements.

The results from the contact surface are presented in the Appendix II. These are von Mises stress, contact pressure, and shear stresses. The shape of the stress distribution is similar for all versions, but the peak stresses are over 25% higher for the node-to-segment versions. This applies for Abaqus as well as for CalculiX.

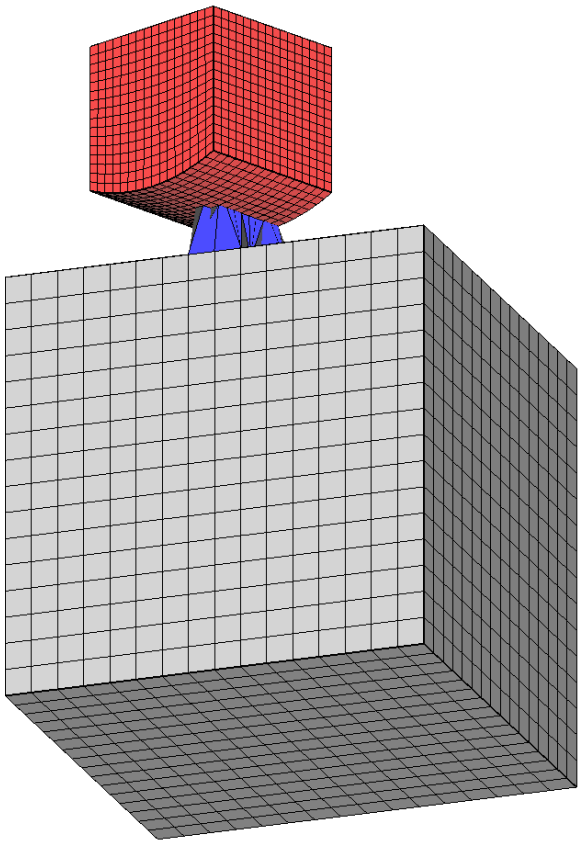


Figure 34. The setup of the punch example which is used for comparison of the results.

Furthermore, the shapes of the distributions of the contact variables are quite similar. However, Gauss Cut version gives significantly higher peak pressure values. This is caused by the dense integration point scheme, because there is most likely contact elements located closer to the edge than in the other versions. Therefore the extrapolated nodal value becomes higher.

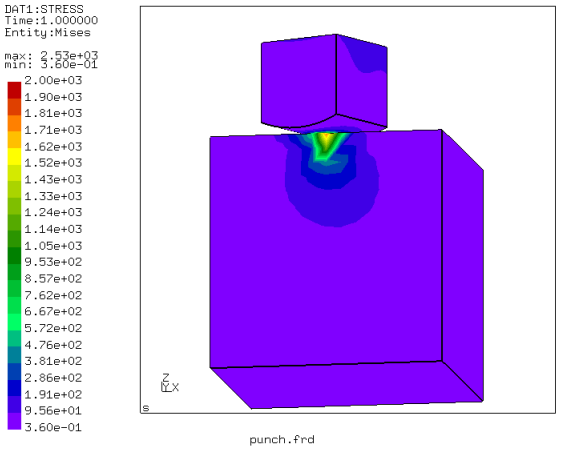


Figure 35a. Von Mises, Abaqus NTS.

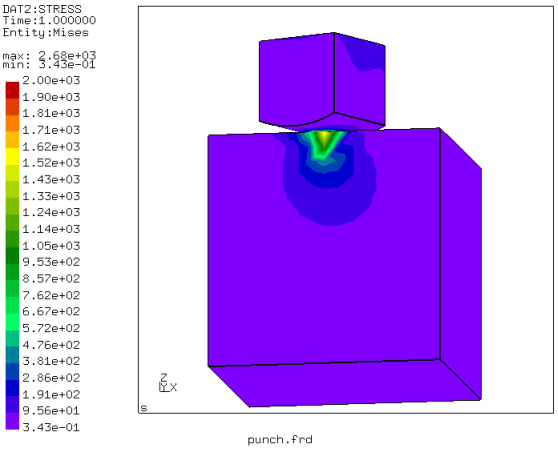


Figure 35b. Von Mises, Nodes.

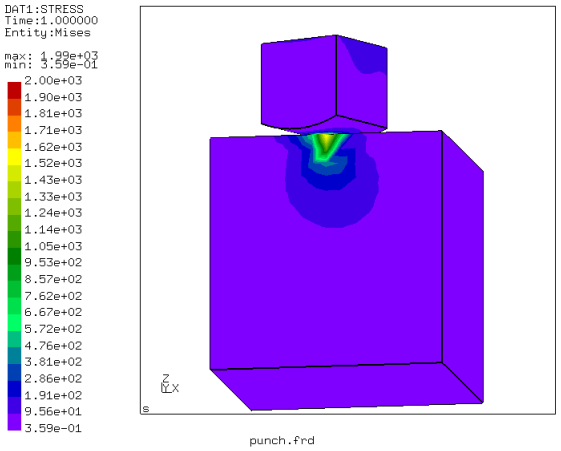


Figure 35c. Von Mises, Abaqus STS.

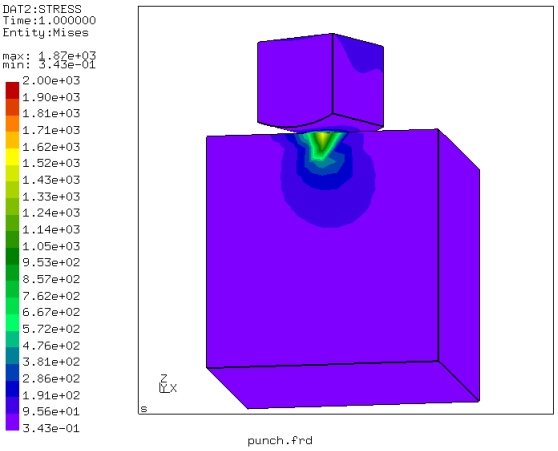


Figure 35d. Von Mises, Linear Gauss.

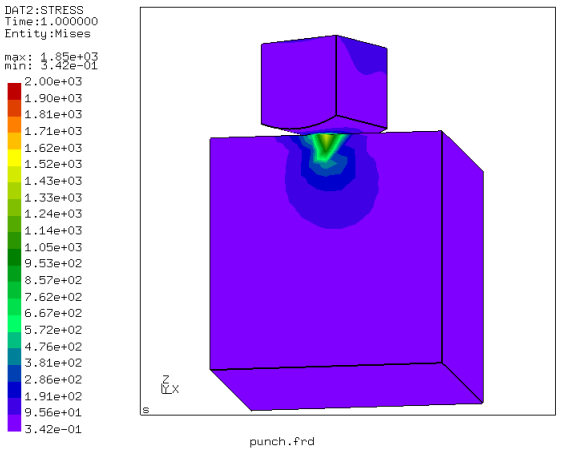


Figure 35e. Von Mises, GC7.

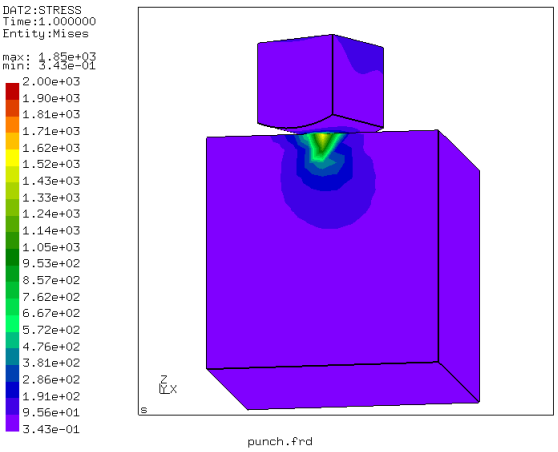
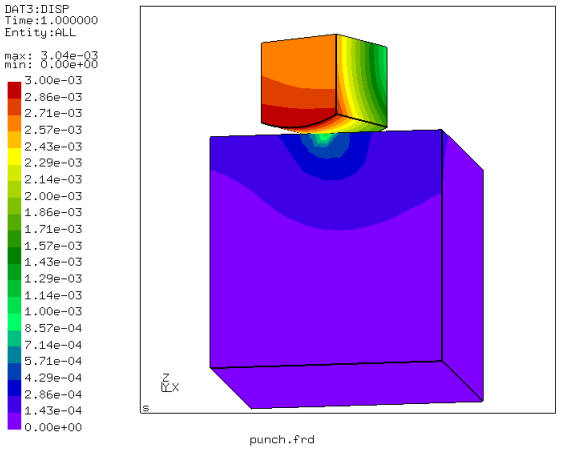


Figure 35f. Von Mises, LGC7.



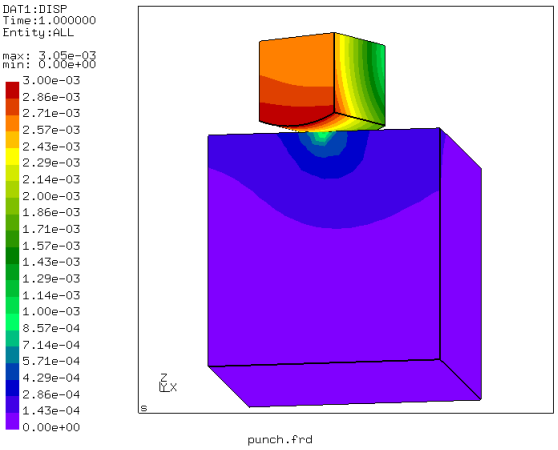
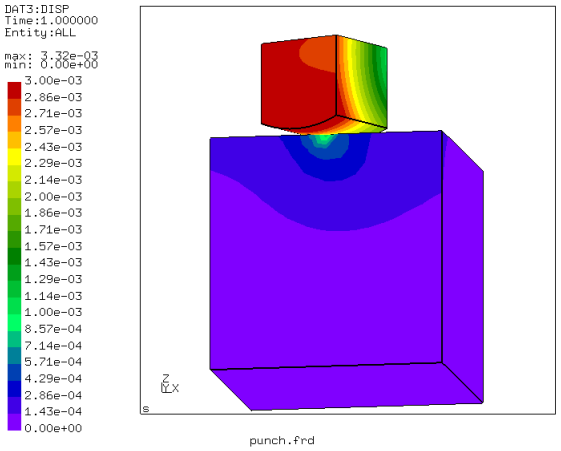


Figure 36a. Displacements, Abaqus NTS. **Figure 36b. Displacements, Nodes.**



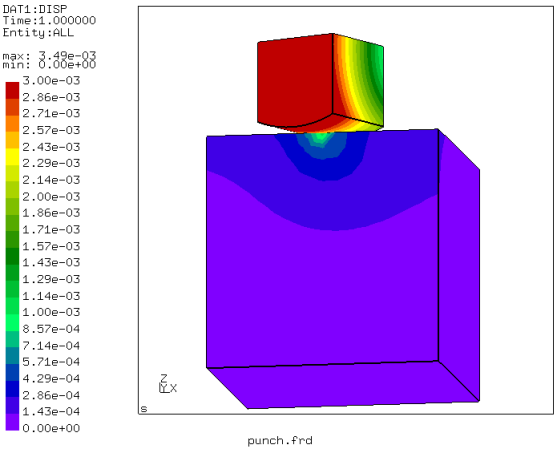
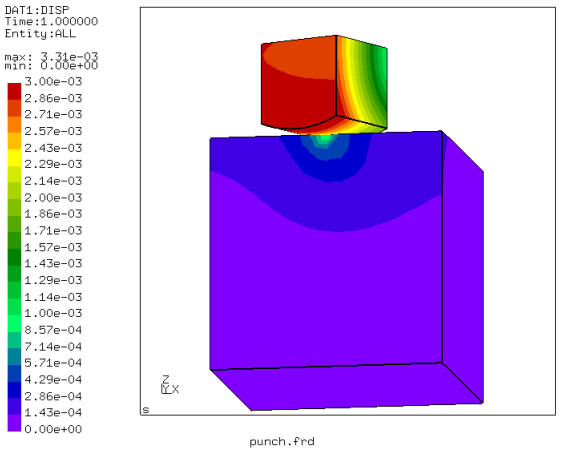


Figure 36c. Displacements, Abaqus STS. **Figure 36d. Displacements, Linear Gauss.**



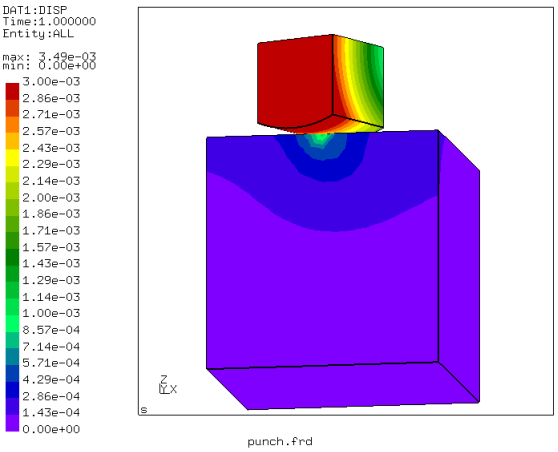


Figure 36e. Displacements, GC7. **Figure 36f. Displacements, LGC7.**

5.2.1 Stability and efficiency

The punch example was also used for the determination of the stability and the efficiency of the segment-to-segment contact formulations. The punch example was run similarly thousand times by varying the location and the rotational position of the upper cube. The comparison is made between eight different contact implementations with linear and quadratic contact surfaces. The results are presented below in the figure 37.

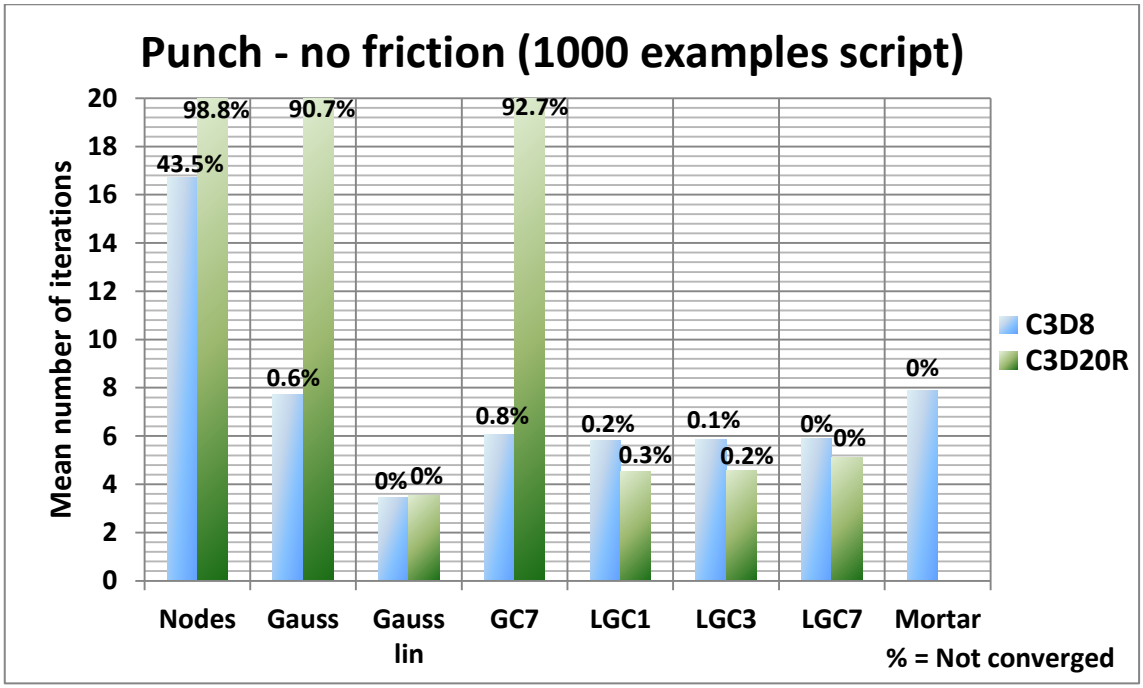


Figure 37. The results for the punch example.

The first thing to note is that, in this table, all three nonlinear versions (Nodes, Gauss, GC7) have actually huge problems with quadratic contact surfaces. Over 90% of the different positions of the upper cube did not converge. With linear elements, all versions except Nodes seem to perform quite well without convergence problems.

For this example, linear versions actually perform even better with quadratic contact surfaces. The Linear Gauss version seems to converge really fast, which is surprising, because it does not settle in line with the other linear segment-to-segment penalty versions. This must be caused by the constant integration point scheme of the Gauss version which seems to suit perfectly for this example. If LGC1, LGC3 and LGC7 are compared, there is again slight indication of the fact that the more integration points, the better the convergence.

5.3 Two beams

The two beams example consists of two vertical beams with a gap between each other (figure 38). The boundary conditions are varied for two different scenarios. The constant pressure loading is applied to the left side surface of the left beam. The plastic material behavior is taken into account. The specifications of the example are shown below.

Basic Parameters

Elastic modulus	$2.08 * 10^5$
Mesh (elements)	2*2*8
Element type	C3D20

Contact parameters

K = 10^6

σ_∞ = 1

(Friction neglected)

Plastic hardening

(Von Mises stress, Equivalent plastic strain)

520, 0
630, 0.005
680, 0.01
710, 0.02
780, 0.05
820, 0.1
880, 0.2
920, 0.35
950, 0.48

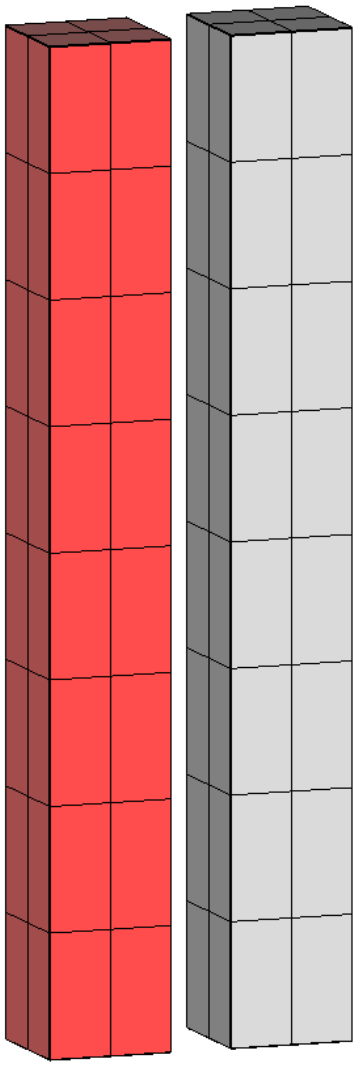


Figure 38. The initial setup of the two beams example.

5.3.1 First setup

The displacements of the first setup are shown in the figure 39. The boundary conditions are such that the both ends of the beams are fixed. The magnitude of the constant pressure loading was increased from 100 to 180 to see how it affects the convergence of the different contact implementations. The table consisting of iterations required to finish the example is shown below.

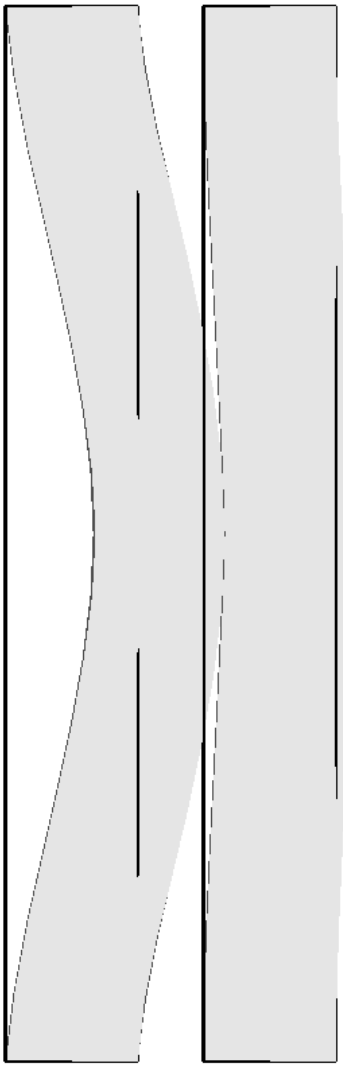


Figure 39. The displacements of the first setup.

Table 5. Convergence table for the first setup.

Loading	Iterations					
	Nodes	Gauss	LGC1	LGC3	LGC7	LGC12
100	141	52	60	12	13	15
120	156	45	46	47	48	49
140	-	10	99	12	66	97
160	-	104	102	99	109	97
180	-	86	72	67	81	100

Nodes version converged for the two lowest pressure loads, but the iterations required were significantly higher compared to the segment-to-segment versions. The segment-to-segment implementations converged much better and more or less with similar amounts of iterations. The results for the uniform pressure loading with the magnitude of 160 are discussed below.

The von Mises stresses are presented in the figures 40a-40f. The interesting thing to note is that the stress distribution given by Abaqus NTS is not symmetric, although the example is perfectly symmetric. However, Abaqus STS gives fully symmetric results. The results from LGC versions are similar independent of the integration point scheme. The peak values are really close to Abaqus. However, Linear Gauss gives somewhat different results and slightly higher peak values.

Displacements and plastic strains are presented in the Appendix III. The displacements given by Abaqus NTS are slightly smaller compared to Abaqus STS. LGC implementations give almost exactly the same displacements than Abaqus STS. The Gauss version gives significantly higher displacements, which seem to be erroneous. The shape of the plastic strain distribution and also the peak values are really close between Abaqus NTS and STS, and LGC versions. The Gauss version gives 25% higher plastic strains.

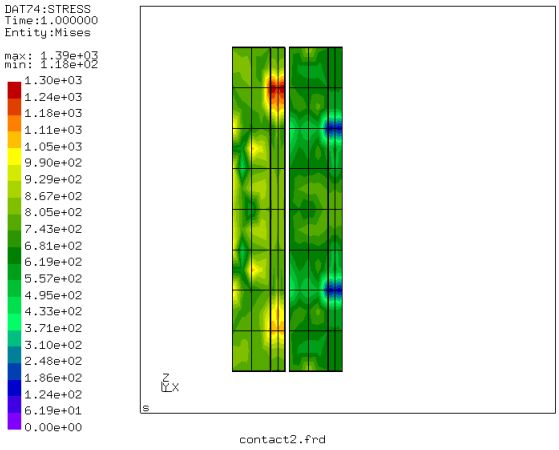


Figure 40a. Von Mises, Abaqus NTS.

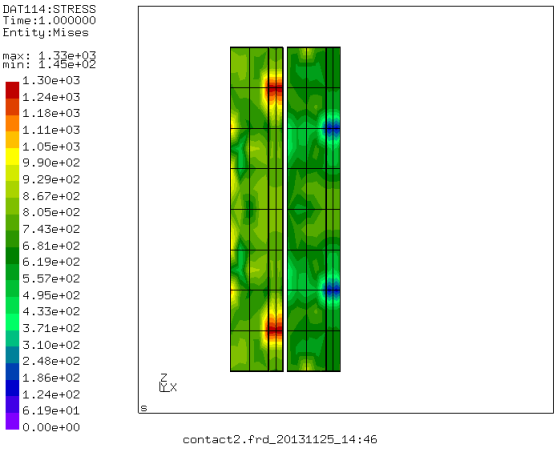


Figure 40b. Von Mises, Abaqus STS.

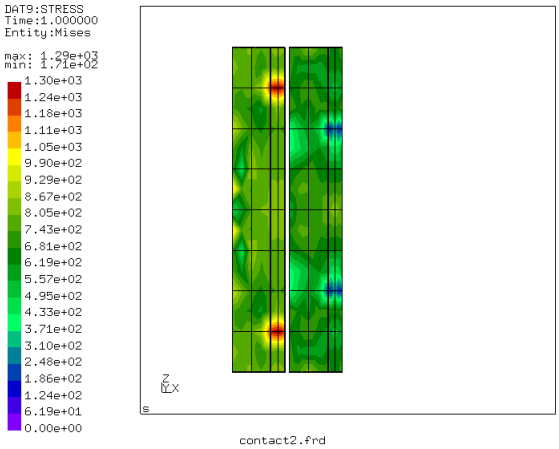


Figure 40c. Von Mises, LGC3.

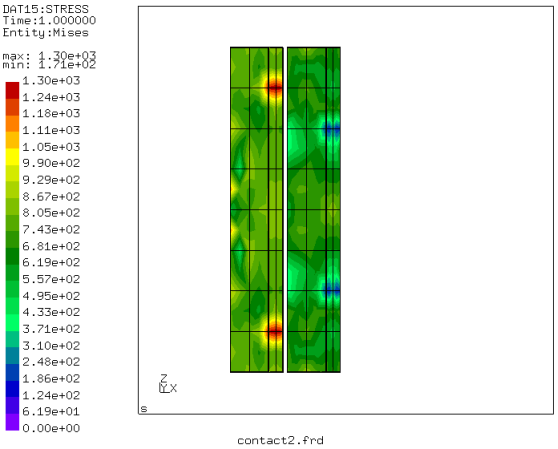


Figure 40d. Von Mises, LGC7.

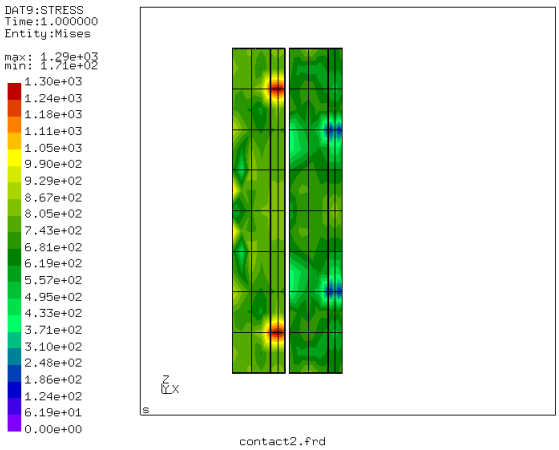


Figure 40e. Von Mises, LGC12.

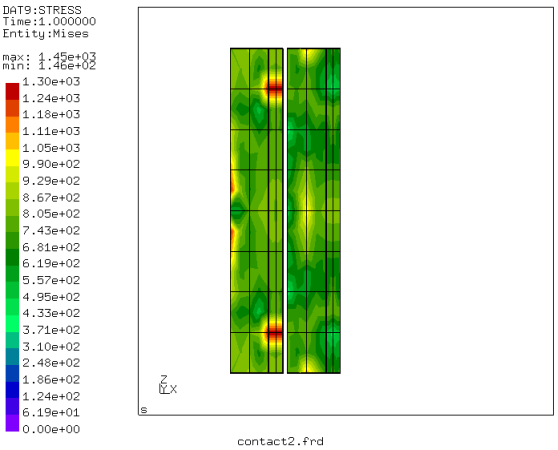


Figure 40f. Von Mises, Linear Gauss.

5.3.2 Second setup

The displacements of the second setup are shown in the figure 41. The boundary conditions are such that the first beam is fixed from the bottom surface and the second beam is fixed from the top surface. The magnitude of the constant pressure loading was increased from 20 to 60 to see how it affects the convergence of the different contact formulations. The table consisting of iterations required to finish the example is shown below.

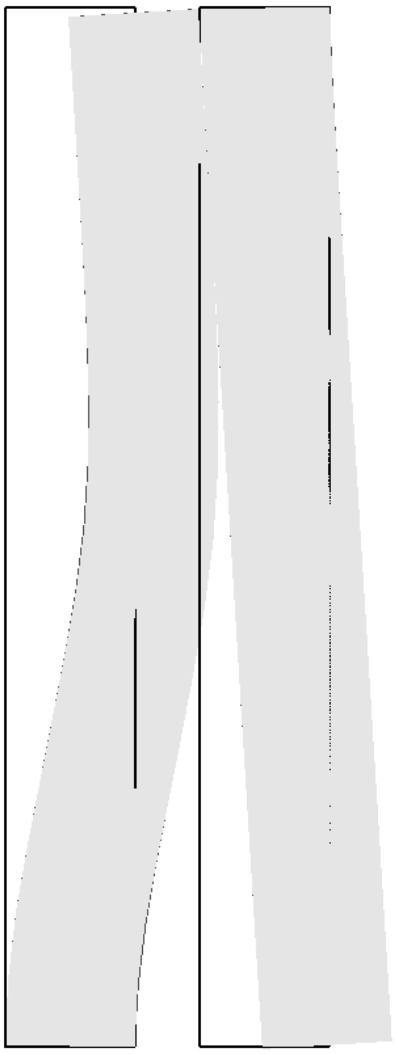


Figure 41. The displacements of the second setup.

Table 6. Convergence table for the second setup.

Loading	Iterations					
	Nodes	Gauss	LGC1	LGC3	LGC7	LGC12
20	165	53	105	-	24	23
30	123	32	-	-	47	41
40	-	66	123	-	147	144
50	-	93	-	90	77	77
60	-	154	-	-	-	-

Nodes version converged again for the two lowest pressure loads, but the iterations required were still significantly higher compared to the segment-to-segment versions. The Gauss version was the most stable version and converged with all pressure loads. LGC1 and LGC3 seem to be slightly unstable for this example. LGC7 and LGC12 were more stable, but still did not converge with the highest pressure load. The results for the uniform pressure loading with magnitude of 50 are discussed below.

The von Mises stresses are presented in figures 42a-42f. Abaqus NTS gives slightly higher stresses compared to Abaqus STS. The peak values of LGC versions are close to each other and similar to Abaqus STS, but the shape of the stress distribution is slightly different. The peak value of the Linear Gauss version correlates with Abaqus NTS, but few high stress spots are missing.

Displacements and plastic strains are presented in the Appendix IV. The displacements given by all versions are completely identical. The shapes of the plastic strain distributions are also really close to each other. However, Abaqus NTS gives 10% higher peak values compared to the other versions.

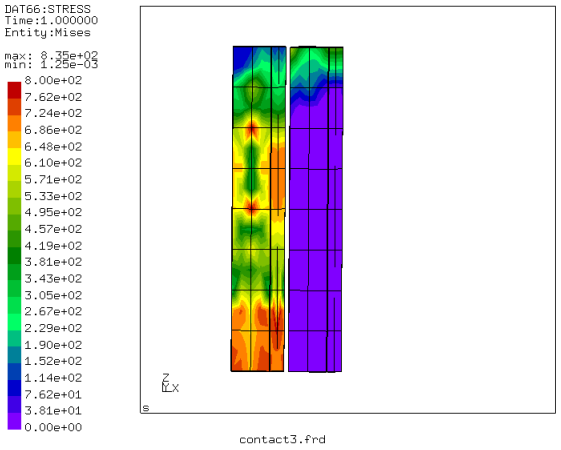


Figure 42a. Von Mises, Abaqus NTS.

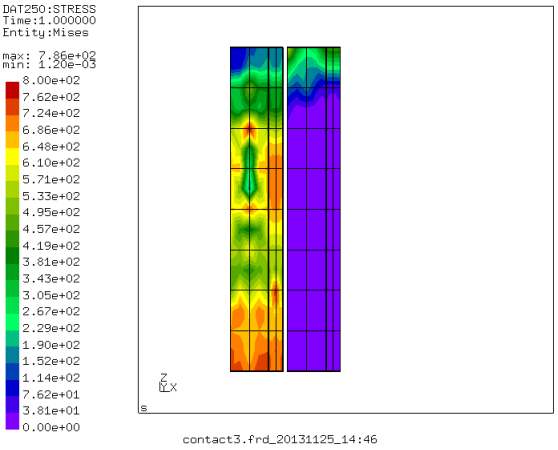


Figure 42b. Von Mises, Abaqus STS.

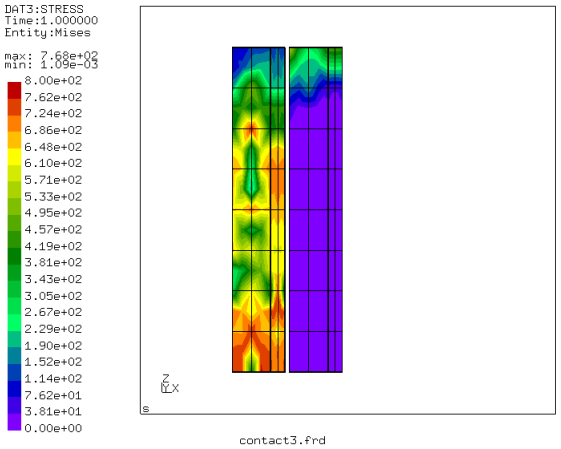


Figure 42c. Von Mises, LGC3.

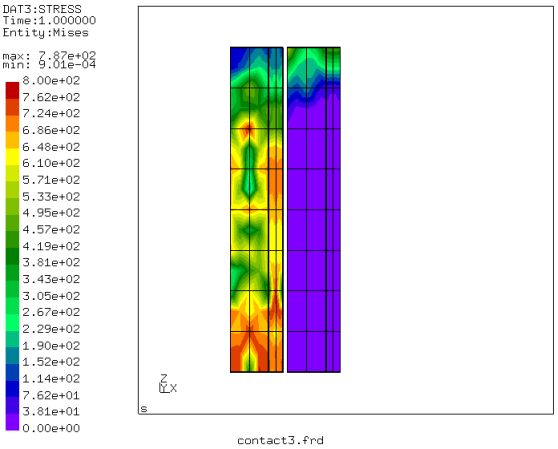


Figure 42d. Von Mises, LGC7.

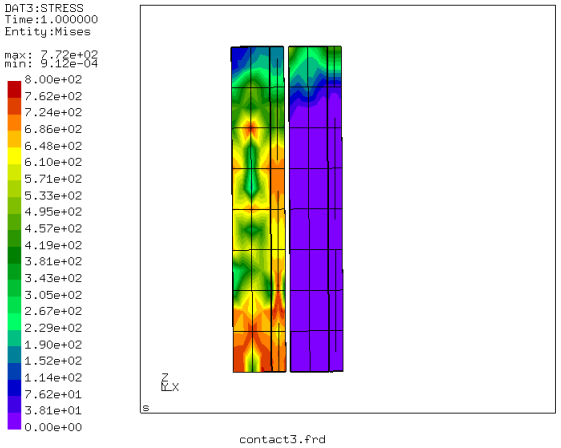


Figure 42e. Von Mises, LGC12.

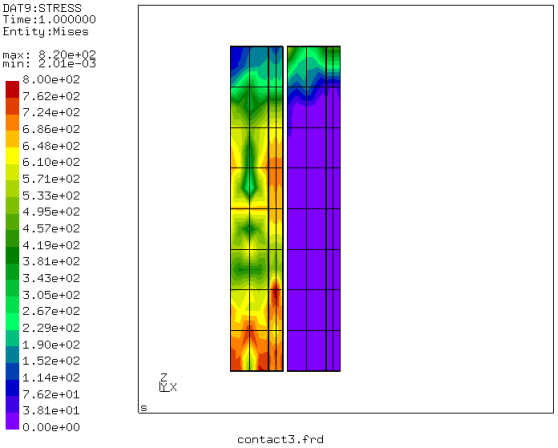


Figure 42f. Von Mises, Linear Gauss.

5.4 Patch test

The patch test is a commonly applied numerical benchmark test for contact problems. The contact patch test resolves whether the correct stress state can be transferred across the contact interface of a given pair of elastic contacting bodies. The meshes of the bodies are nonconforming and typically none of the nodes of the two contact surfaces coincide. (Taylor, & Papadopoulos, 1991; Baig, 2006)

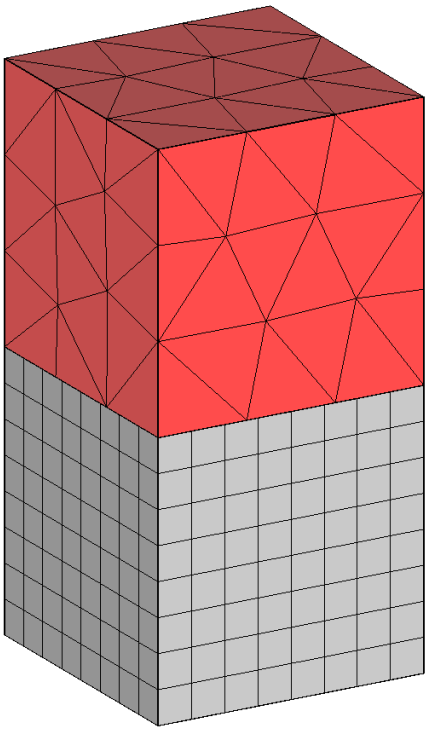


Figure 43. The patch test example.

The applied model is such that two geometrically identical cubes are placed one upon the other (figure 43). The top surface of the upper cube is subjected to a uniform pressure distribution of unit magnitude. Poisson's ratio is set to zero, which makes sure that the only real nonzero stress state is the vertical compressive stress of unit magnitude. The mesh of the upper cube is coarse and consists of quadratic tetrahedral elements (C3D10), whereas the mesh of the lower cube is much finer and consists of quadratic hexahedral elements (C3D20).

The boundary conditions are similar to the cube and the punch examples. The top surface of the upper cube is fixed in x- and y-directions. The bottom surface of the lower cube is fixed in all directions (x, y and z). The spring constant is set to $K = 10^7$ and the value for tension at large clearances is $\sigma_\infty = 1$. Friction is neglected. The bottom surface of the upper cube is defined as the slave surface, which means that the integration points for the node-to-segment formulations as well as the Gauss segment-to-segment formulation are determined by these slave faces. The integration points of the Gauss Cut formulation are determined based on the cutting of the slave and the master faces hence the integration points do not depend on the choice of the slave surface. The cutting of the slave and the master surface is shown in the figure 44.

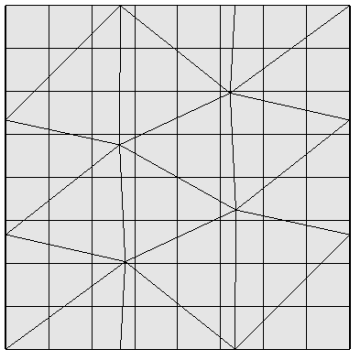


Figure 44. The cutting of the contact surfaces.

The results are compared between Abaqus NTS, Abaqus STS, Nodes (2.6.2), Linear Gauss and Linear Gauss Cut one-, three-, seven- and twelve-point schemes. The maximum values for the contact pressures as well as the maximum and the minimum values for the normal stresses in the z-direction are shown in the table 7. Additionally, the stress distribution for the z-direction is presented in the figures 46a-46f for all contact formulations except for LGC3 and LGC12. The results between LGC versions of three-, seven- and twelve-point schemes were however perfectly identical. Re-meshing was not used for any CalculiX version so that the results are perfectly comparable.

Table 7. The results of the patch test.

	Abaqus NTS	**Abaqus STS**	**Nodes**	**Linear Gauss**	**LGC1**	**LGC3**	**LGC7**	**LGC12**
σ_{zz} (min)	-3.57	-1.11	-3.99	-2.52	-1.01	-1.00	-1.00	-1.00
σ_{zz} (max)	0.625	-0.881	0.356	1.21	-0.918	-0.999	-0.999	-0.999
CPress (max)	1.53	1.01	1.73	1.68	1.06	1.00	1.00	1.00

The first thing to note from the results is that the node-to-segment formulations, but also the Gauss formulation perform quite poorly for the patch test. The reason for the Gauss version's bad success is caused by the integration points which are determined by the slave surface. The mesh of the slave surface is much coarser than the mesh of the master side, and the amount of integration points created by the Gauss version is simply not enough.

The perfect results would be simply uniform normal stress of minus one in the z-direction and uniform contact pressure of one. Nodes version gives the worst results; the difference between the minimum and the maximum normal stress is approximately 4.35. The maximum contact pressure was also the highest: 1.73. Abaqus NTS did not perform much better, as the difference between the minimum and the maximum normal stress is still over four. The maximum contact pressure is 1.53. The Gauss version gives slightly better outcome, but the results are still more or less similar compared to the node-to-segment formulations.

All other segment-to-segment formulations performed much better. Abaqus STS gives the worst results of these formulations; the difference in the vertical normal stress is about 0.23 and the maximum contact pressure is 1.01. LGC1 version performed quite similar to Abaqus STS. LGC versions with denser integration point schemes give almost perfect results; the error in the vertical normal stress is only one per mille, which is really close to the perfect uniform stress distribution. The pressure is uniform and exactly one, within the accuracy of three significant figures.

It seems that the way of creating integration points in the Gauss Cut version has significant advantages in case the meshes of the slave and the master surface are not similar. Additionally, the results are always the same independent of the choice of the slave and the master surface. On the other hand, the problem of the Gauss integration point scheme becomes very clear. For the Gauss Cut formulation, even one-point scheme seems to give quite good results. However, better results can be achieved by denser integration point schemes. For the patch test, three-point scheme seems to be enough to produce close to ideal results. The contact pressures for Nodes and LGC7 version are presented in the figures 45a-45b to illustrate the advantage of the segment-to-segment formulation.

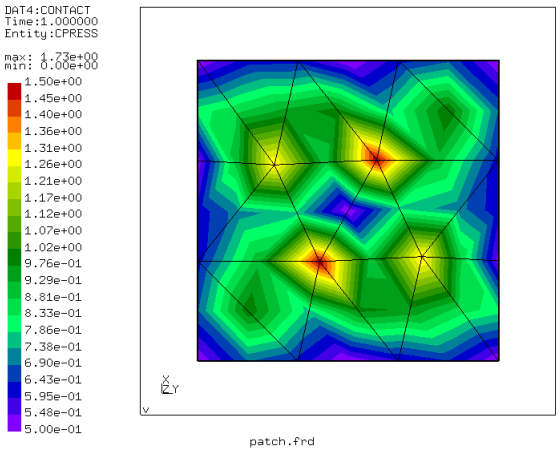


Figure 45a. Contact Pressure, Nodes.

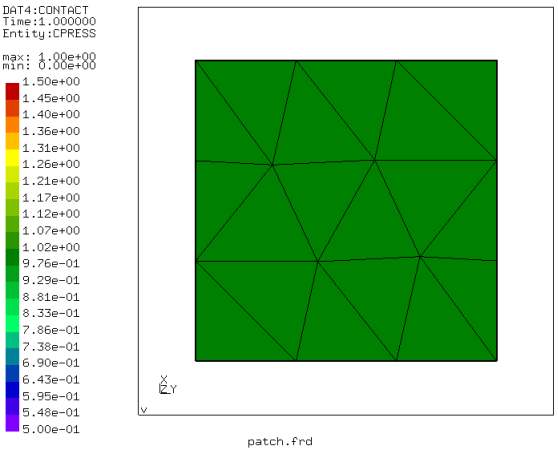


Figure 45b. Contact Pressure, LGC7.

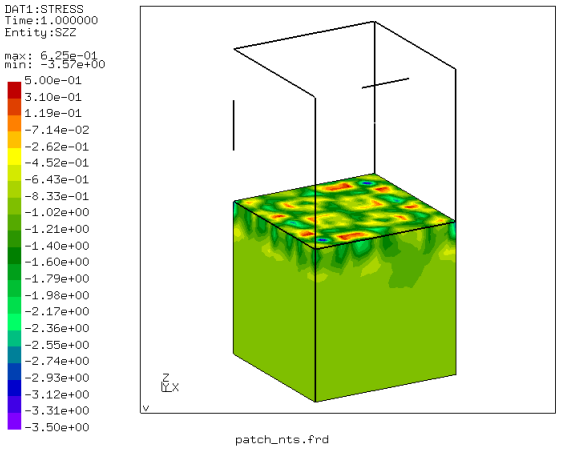


Figure 46a. σ_{zz}, Abaqus NTS.

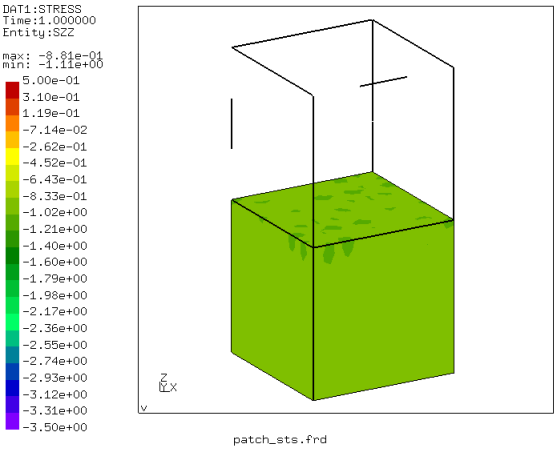


Figure 46b. σ_{zz}, Abaqus STS.

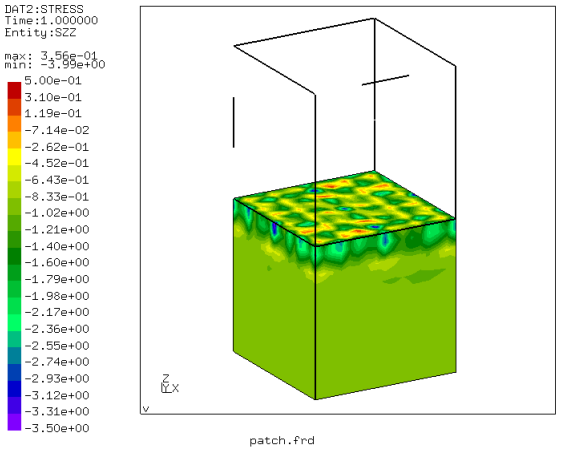


Figure 46c. σ_{zz}, Nodes.

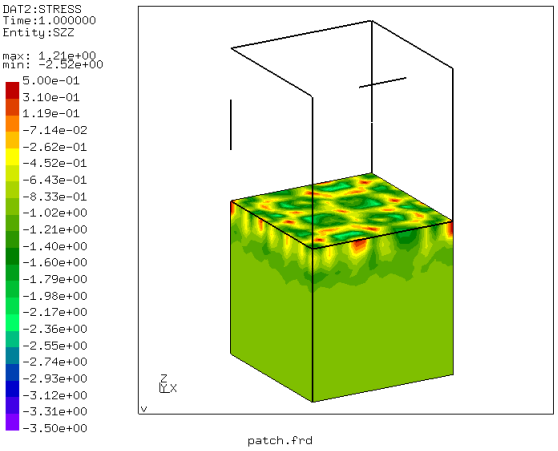


Figure 46d. σ_{zz}, Linear Gauss.

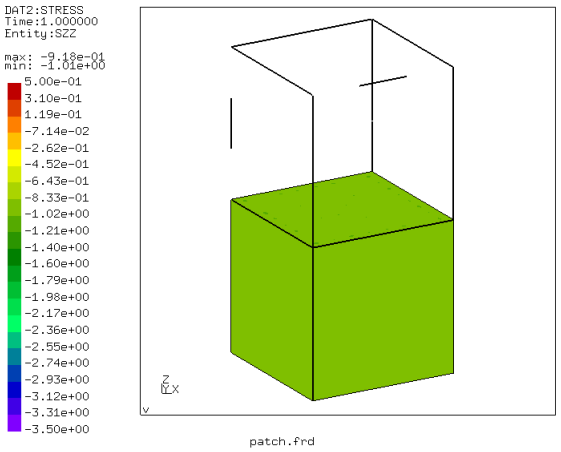


Figure 46e. σ_{zz}, LGC1.

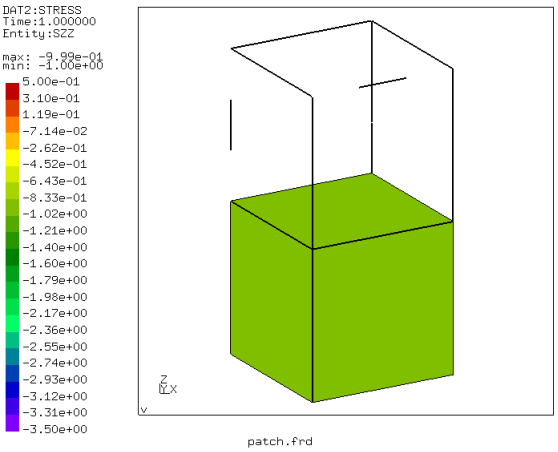


Figure 46f. σ_{zz}, LGC7.

5.5 Patch test - tied contact

The same patch test geometry was used to compare tied contact. Tied contact means that the contact surfaces are "glued" together and cannot have any displacements relative to each other. Currently in CalculiX tied contact is modelled by introducing multiple point constraints (MPC's) for the nodes located at the contact surfaces. However, since the contact constraints are defined only for the nodes, similar results compared to the node-to-segment formulations are expected.

LGC7 contact formulation was modified to be suitable for tied contact calculation. This was done by setting c_0 to a very high number, which means that a contact element is always created for each integration point provided that there exists an opposite master surface. The truly linear pressure-overclosure -relationship is already able to provide compressive as well as tensional contact forces. Additionally, the friction coefficient was set to a very high number $\mu = 10^{30}$ and the tangent of the stick range was set equal to the spring coefficient $\Lambda = K = 10^{7}$. This arrangement makes sure that the displacements to all directions are equally constrained and no tangential slip occurs.

The applied boundary conditions for the example were not changed from the previous section. However, the sign of the uniform pressure distribution was changed such that the only real nonzero stress state is the vertical tensional stress of unit magnitude. In other words, the tied contact formulation must now be able to keep the cubes together.

The results for the present tied contact formulation in CalculiX as well as the modified LGC7 formulation are shown in the figures 47a-47b. The results for the MPC based tied contact formulation are similar to the results of Nodes contact formulation presented in the previous section. This was expected as the contact constraints for both cases are only satisfied at the nodal locations. The main difference is actually the spring constant, which does not exist for tied contact formulation. In other words, this corresponds to an infinite spring constant for Nodes formulation. The results for the modified LGC7 formulation are almost ideal similarly to the previous section.

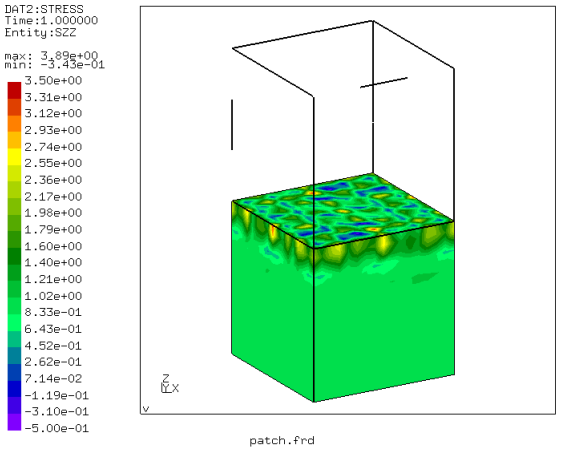


Figure 47a. σ_{zz}, MPC's – tied contact.

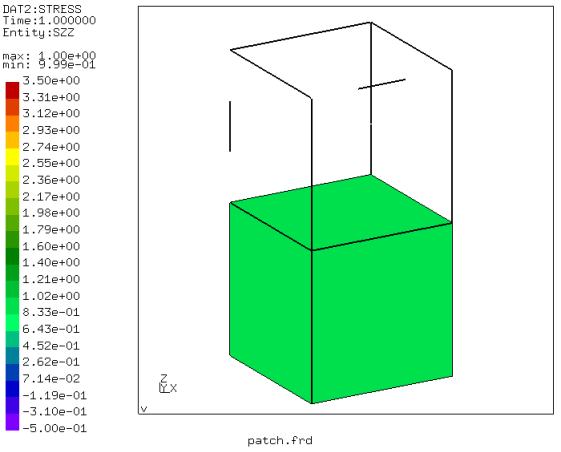


Figure 47b. σ_{zz}, modified LGC7.

6 Real models

Many models of turbine engine parts were used to test the new contact formulations. The models are mostly related to the attachments of turbine blades and stator vanes, but also contact between casing parts is considered. Important results are accuracy, convergence and computational time. Accuracy is determined by comparing the results to Abaqus, convergence by the amount of steps converged and computational time in terms of total iterations needed by the solver.

The first part of this section compares the convergence of the examples run by different contact implementations. The comparison is made between Abaqus NTS (node-to-segment) and STS (segment-to-segment), Nodes (2.6.2) (node-to-segment penalty), Linear Gauss, and Gauss Cut nonlinear and linear version.

In the second part, the two promising contact formulations are compared to Abaqus; LGC3 and LGC7. If the CalculiX node-to-segment penalty contact method converged, those results are compared as well. The quantities compared are typically worst principal stress, contact pressure and possible contact shear stresses. The maximum values of the variables presented in this chapter are normalized such that the maximum value given by Abaqus becomes one. The unit system used throughout this section is {mm, N, s, K}. If the units are not specified within the text, the aforementioned units may be assumed.

6.1 Convergence and computational time

The convergence characteristics are shown in the table 8. The first column consists of different turbine engine part models which were run by all different contact implementations. The list consisting of the explanations of the models is given below.

CA: Contact between casing parts
CV: Contact between a casing and stator vanes
DA: Contact between dampers and blade feet
DB/Y: Contact between a disk and a blade foot
FL: Contact between a bolt and casing parts

As one may note from the table attached below, the convergence of Nodes (node-to-segment penalty method) is not very impressive. The second thing to note is that the nonlinear versions of the Gauss Cut contact formulation did not perform that well either. Generally speaking, the iterations in Nodes version are significantly faster, however smaller increments are usually needed which makes the segment-to-segment implementations even faster. If constant increment size is applied, then Nodes version would be faster.

Table 8. Convergence of the different contact formulations in terms of iterations.

Model	Nodes 2.6.2	Abaqus NTS	Abaqus STS	Linear Gauss (rem)	Linear Gauss (no rem)	GC7 (rem)	GC7 (no rem)	LGC7 (rem)	LGC7 (no rem)
CA1	-	105	115	-	-	-	-	121	136
CV1	-	72	531	364	38	-	-	38	41
CV3	256	18	18	21	18	510	-	22	23
CV4	-	36	31	27	23	-	-	33	45
DA	-	9	6	8	7	8	9	8	8
DB1	-	22	26	8	7	-	225	9	9
DB2	-	35	8	15	19	-	-	13	15
FL	-	65	89	206	117	763	-	103	110
Y1	-	152	7	-	9	-	-	-	11
Y140	67	5	6	10	10	10	10	7	7
Y175	179	6	6	10	10	11	11	9	9
Y4	13	12	16	17	15	12	16	18	11

Linear Gauss and Gauss Cut implementations performed really well without significant difference between quadratic and re-meshed linear contact surfaces. Actually, the Y1 example converged only with the version not using re-meshing. Because there was no intention towards using re-meshing, the further experiments for comparing different integration point schemes were conducted only to the version without re-meshing. It seems that the number of integration points does not have a significant effect on the required iterations.

With the Linear Gauss Cut version, all examples in the table 8 converged without re-meshing. The Gauss version performed quite well also, but one example did not converge. In terms of iterations CalculiX seems to be quite close to Abaqus. For some examples, the new segment-to-segment penalty formulations converge in even less iterations than Abaqus. For further comparison between the best versions, another table is attached (table 9). This table combines the results only from Nodes, Abaqus NTS/STS, and Linear segment-to-segment penalty implementations with different integration point schemes and with no re-meshing.

Table 9. Convergence of the different contact formulations in terms of iterations.

Model	Nodes 2.6.2	Abaqus NTS	Abaqus STS	Linear Gauss (no rem)	LGC1 (no rem)	LGC3 (no rem)	LGC7 (no rem)	LGC12 (no rem)
CA1	-	105	115	-	129	140	136	137
CV1	-	72	531	38	41	39	41	44
CV3	256	18	18	18	23	24	23	24
CV4	-	36	31	23	42	40	45	39
CC1	-		-	-	133	126	112	-
DA	-	9	6	7	7	8	8	8
DB1	-	22	26	7	9	9	9	9
DB2	-	35	8	19	13	16	15	16
FL	-	65	89	117	100	111	110	112
FL2	-	21	21	32	34	35	35	31
PO	-	-	-	32	30	30	30	30
Y1	-	152	7	9	8	-	11	12
Y140	67	5	6	10	7	7	7	7
Y175	179	6	6	10	8	8	9	9
Y4	13	12	16	15	11	11	11	11

The second table is quite similar to the first one, but few extra examples are introduced and more integration point schemes for the Linear Gauss Cut version are added. Furthermore, the nonlinear Gauss Cut versions as well as the versions with re-meshing are dropped, because they were not much of an interest anymore. The new model types are introduced below.

CC: Contact between a c-clip, a casing and a seal
PO: Contact between a disk, a retainer and a blade foot.

The CC1-example converged only by LGC1, LGC3 and LGC7 versions. The example is big and challenging due to the multiple contact areas. LGC7 converged fastest. FL2-example converged steadily with all versions. PO-example converged quite well with all segment-to-segment penalty implementations. However, surprisingly it did not converge with Abaqus NTS or STS.

6.2 Disk – blade foot 1 (Y4)

The first example has two contact zones connecting the blade and the disk together. The two contact zones are marked in red color in the figure 48. This example uses cyclic symmetry option, which means that the calculation is only performed for one sector of the whole turbine. This significantly reduces computational time and the amount of required memory. The mesh consists of quadratic tetrahedral elements. Temperature and centrifugal loadings are applied to simulate the spinning of the model. The contact parameters are presented below:

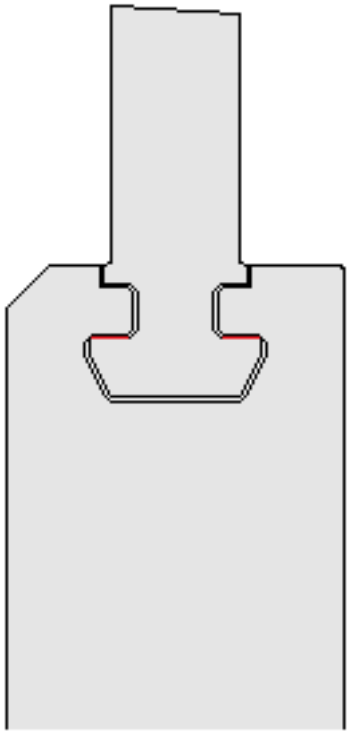


Figure 48.
A sketch of the Y4 - example.

Contact parameters

K	=	10^7
σ_∞	=	3
μ	=	0.75
Λ	=	$5*10^4$

The worst principal stress of the left contact zone is presented in the figures 49a-49d. The first thing to notice is that the stress distribution of Nodes version clearly differs from the others. The contact zone, which is in the middle of the picture, is rough, which is actually caused by re-meshing. Similar results were also obtained by the new segment-to-segment penalty formulations, when re-meshing of the quadratic contact surfaces was conducted. After all, the results from LGC versions are really close to Abaqus STS.

The contact pressures and the shear stresses are presented in the Appendix V. The non-smooth results for Nodes version can be noticed also for the pressure and the shear stress distributions. The contact pressures of LGC versions are smoother, but the peak pressures at the boundary areas are about 50% higher compared to Abaqus. The maximum pressures for Nodes version are about 10% lower than the pressures given by Abaqus.

The shapes of the distributions of the shear stresses are quite similar. The maximum values between all CalculiX versions are quite similar. However, these values are only about 35% the values of Abaqus.

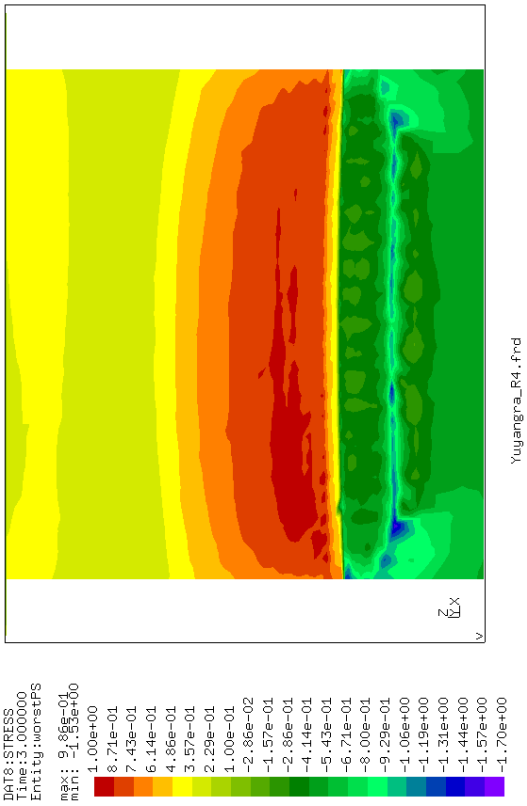


Figure 49a. WorstPS, Nodes.

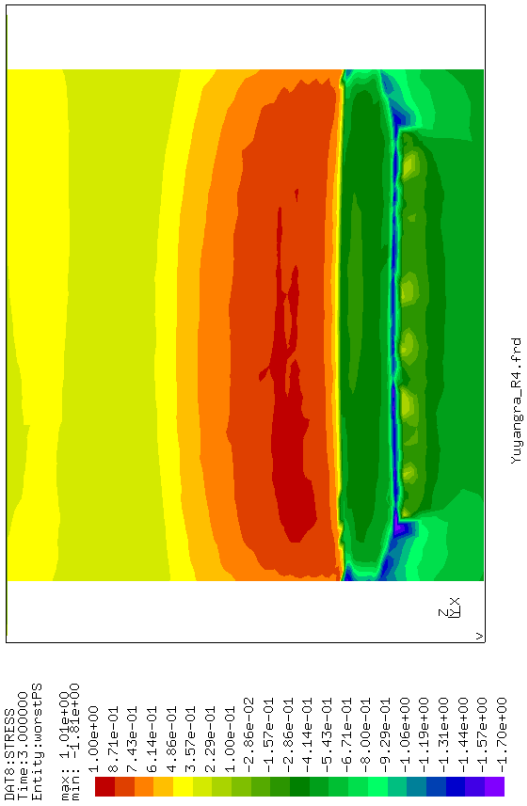


Figure 49b. WorstPS, LGC7.

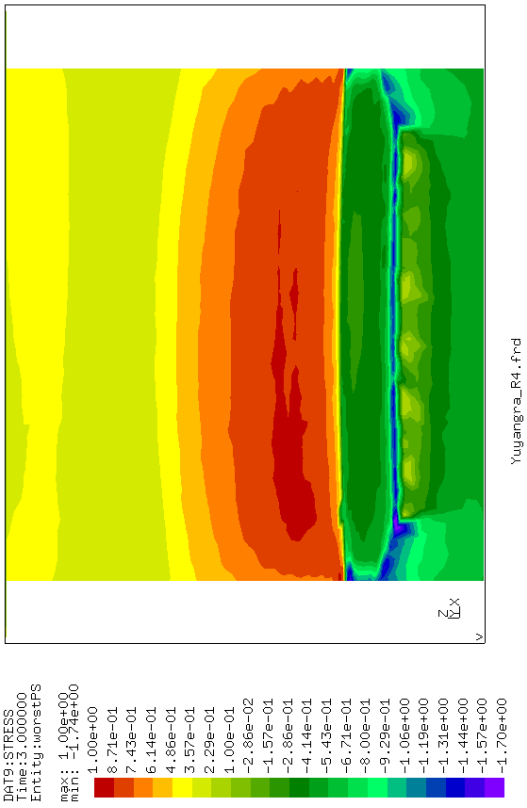


Figure 49c. WorstPS, Abaqus STS .

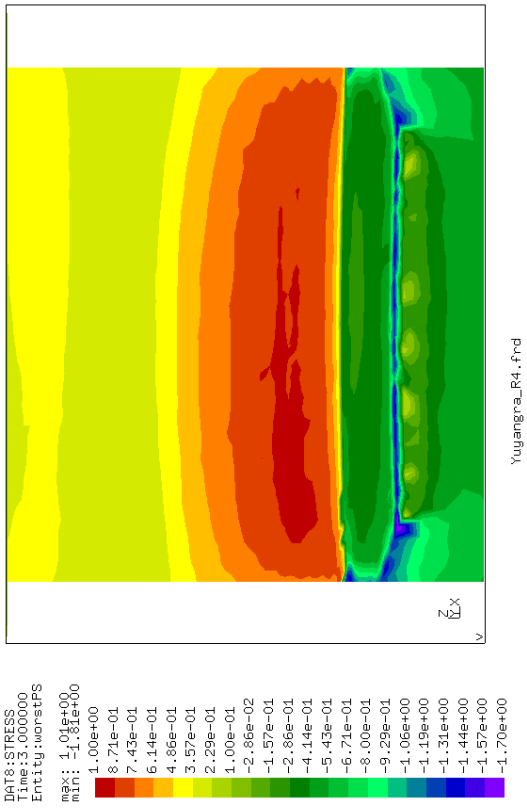


Figure 49d. WorstPS, LGC3.

6.3 Disk – blade foot 2 (DB1)

This example has four contact zones connecting the blade foot and the disk together. The four contact zones are marked in red color to the figure 50. The example uses cyclic symmetry option. The mesh of the blade consists of quadratic tetrahedral elements, whereas the disk is constructed from linear brick elements. Heterogeneous temperature distribution, centrifugal loading, and also concentrated loading to the blade are applied within one step to simulate the spinning of the model. The contact parameters are presented below:

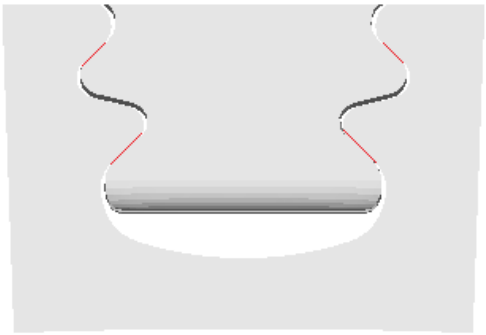


Figure 50. A sketch of the contact zones of the DB1 –example.

Contact parameters

K	=	10^7
σ_∞	=	3
μ	=	0.75
Λ	=	$5*10^4$

The worst principal stress distribution from one side of the example is shown in the figures 51a-51d. Nodes version gives again slightly different results. The shape of the stress distribution is still a little rough and stresses are also slightly lower compared to all segment-to-segment formulations. The results from LGC versions are really close to Abaqus STS.

The pressure and the shear stress distributions are presented in the Appendix VI. In general, the contact zones are quite narrow for this example. However, both LGC versions give slightly wider contact zone. Here, Nodes version is actually closer to Abaqus. The maximum pressure values from all CalculiX versions are a little lower than the values given by Abaqus. For the first shear stress direction, CalculiX versions give significantly higher peak values. However, for the second direction, the CalculiX values are a little lower.

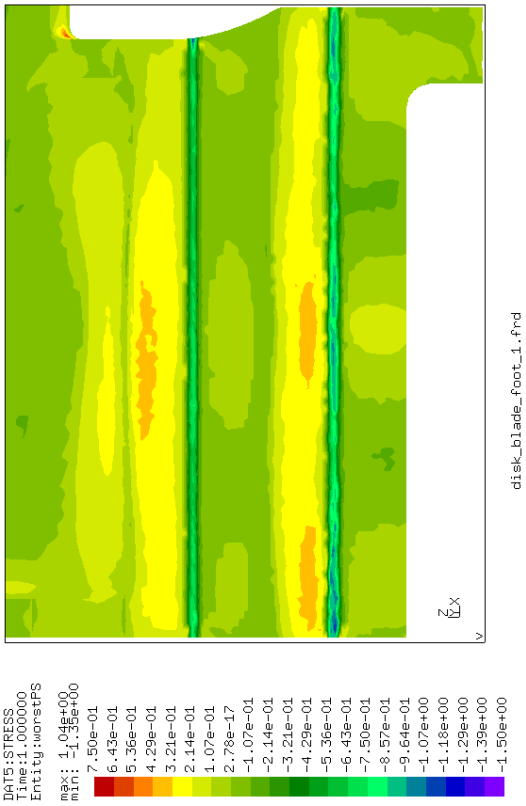


Figure 51a. WorstPS, Nodes.

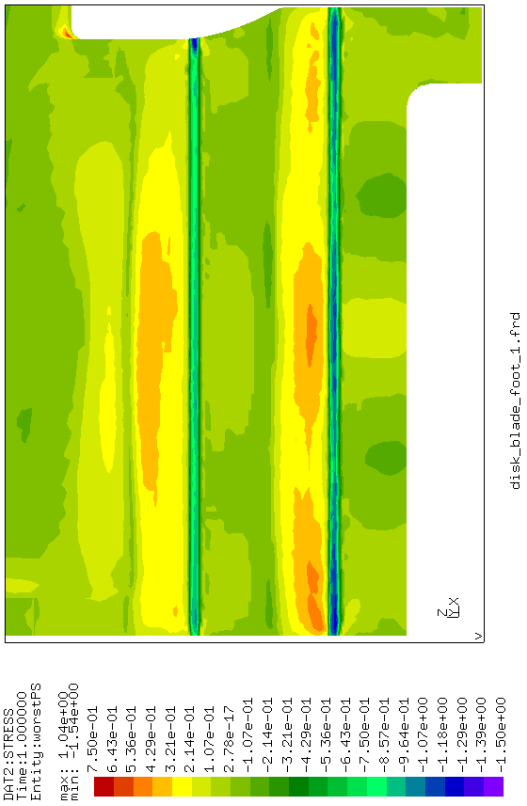


Figure 51b. WorstPS, LGC7.

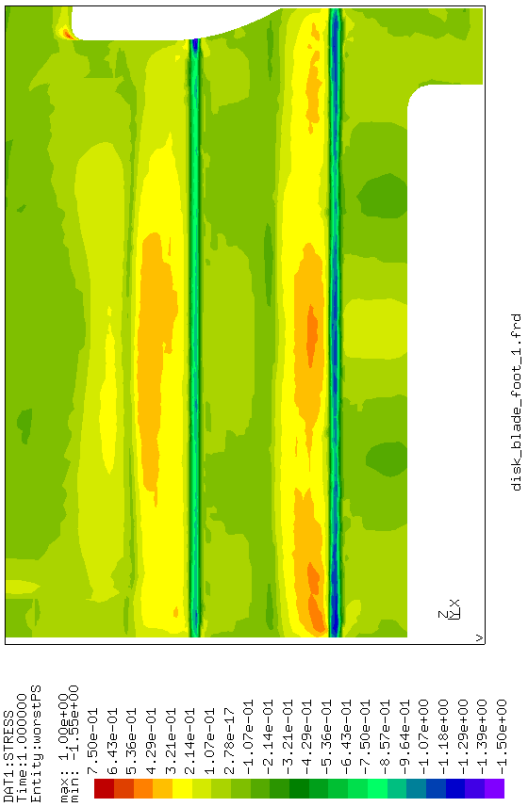


Figure 51c. WorstPS, Abaqus STS .

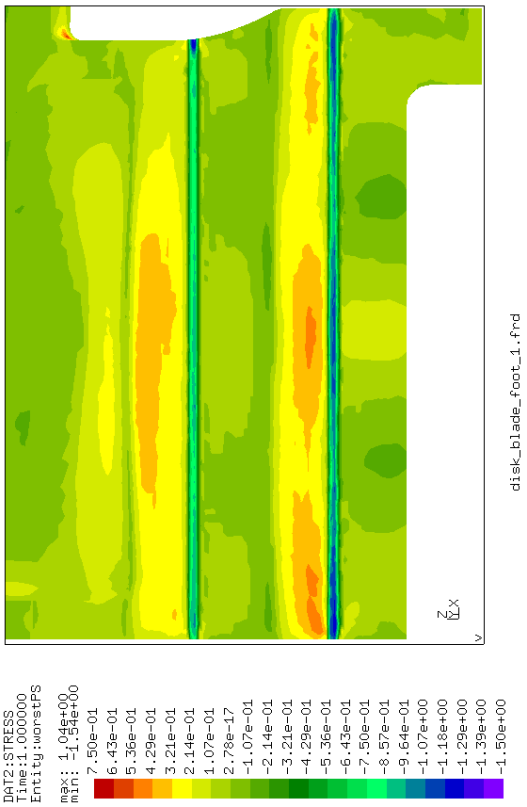


Figure 51d. WorstPS, LGC3.

6.4 Flange 1 (FL1)

In the Flange 1 example, two parts are attached together by one bolt. The example has multiple contact zones. The contact zones are marked in the figure 52 in red color. The mesh consists of quadratic brick elements. The boundary conditions are such that the model is fixed from the left bottom part, and a force towards positive X-direction is applied to the right upper part of the example. Additionally, the surface pointing outwards from the figure 52 is not allowed to have displacements in the Y-direction. The contact parameters are presented below:

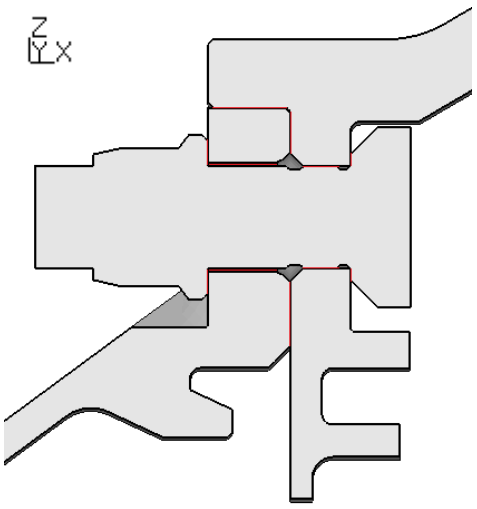


Figure 52. A side view of the FL1 –example.

Contact parameters

K	=	10^7
σ_∞	=	3
μ	=	0.75
Λ	=	$5*10^4$

The worst principal stress distribution from the "open" side of the example is shown in the figures 53a-53c. The same stresses for the bolt are given in the figures 54a-54c. Nodes version did not converge for this example so it is omitted. The worst principal stresses given by LGC versions for the whole joint, and also for the bolt are really close to Abaqus. The peak values for LGC versions are approximately 10% higher than the values given by Abaqus.

The pressure and the shear stress distributions are presented in the Appendix VI. The contact zones are relatively similar, but actually LGC versions seem to give smoother results. The maximum values of the pressures for both CalculiX versions are a little over 10% lower than the value given by Abaqus. The shear stresses given by Abaqus are really rough, but the distribution is quite similar to CalculiX. The maximum values given by both LGC versions are more than twice as high as the values given by Abaqus.

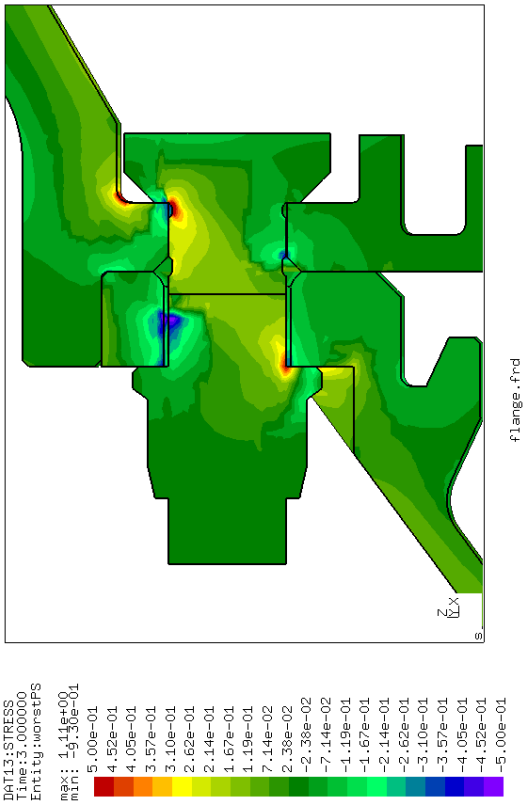


Figure 53a. WorstPS, LGC7.

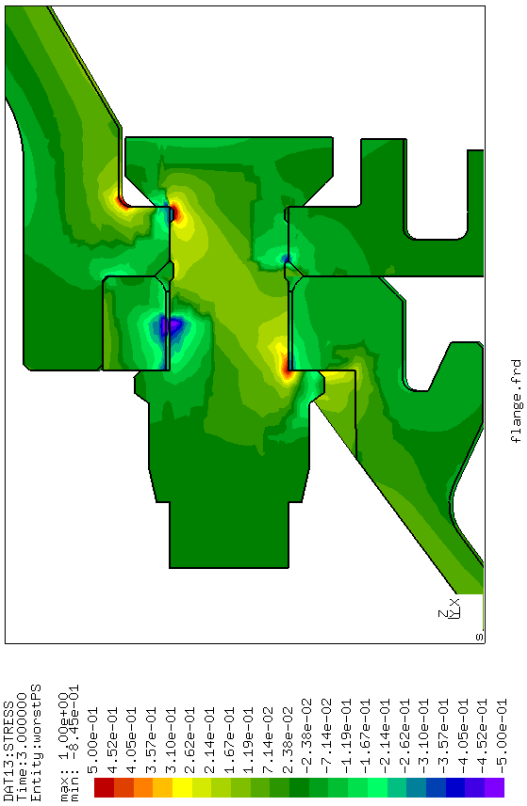


Figure 53b. WorstPS, Abaqus STS.

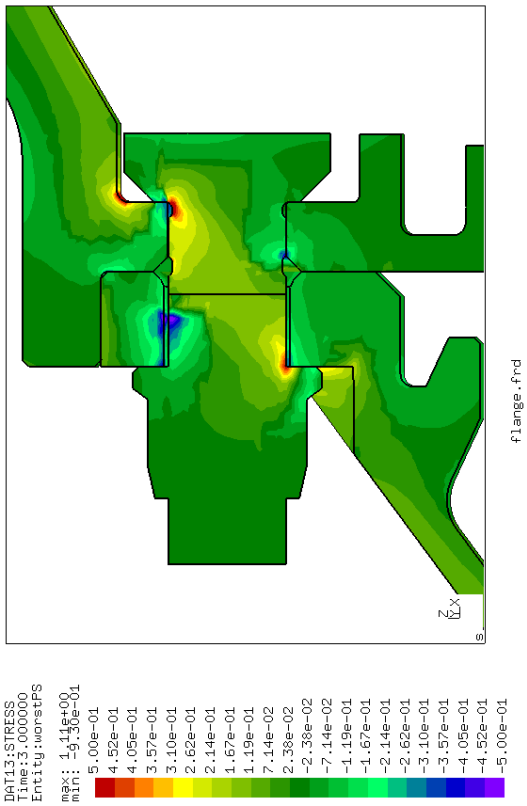


Figure 53c. WorstPS, LGC3.

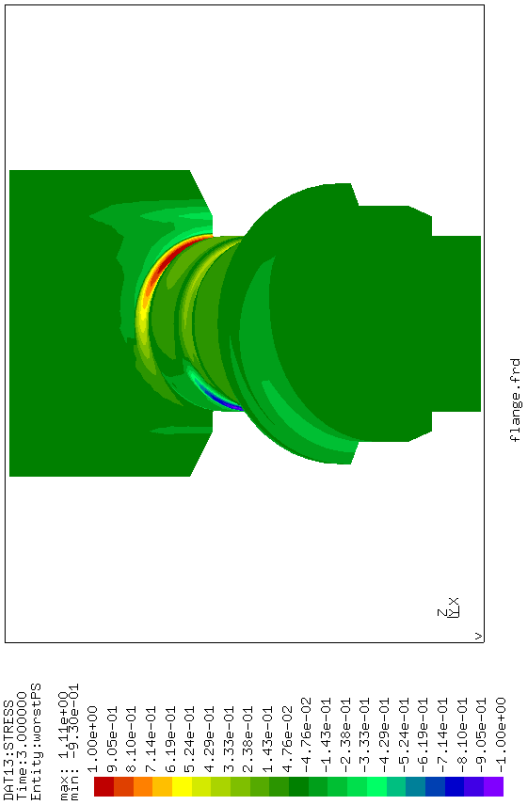


Figure 54a. WorstPS, LGC7.

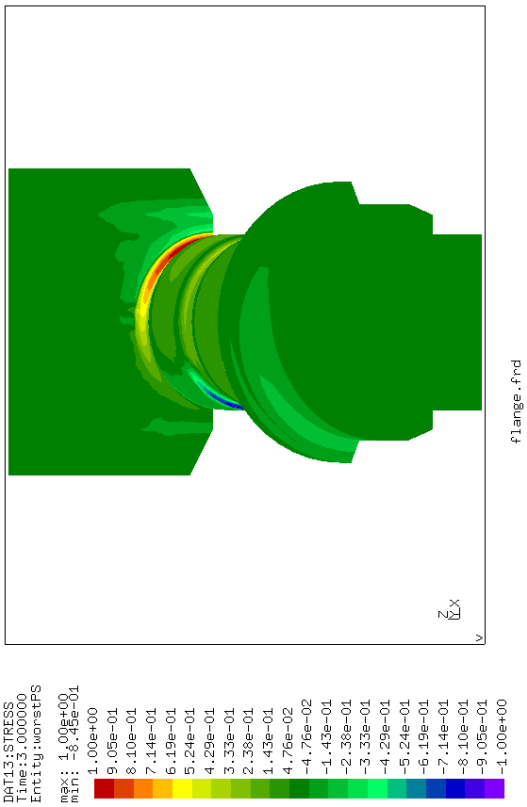


Figure 54b. WorstPS, Abaqus STS.

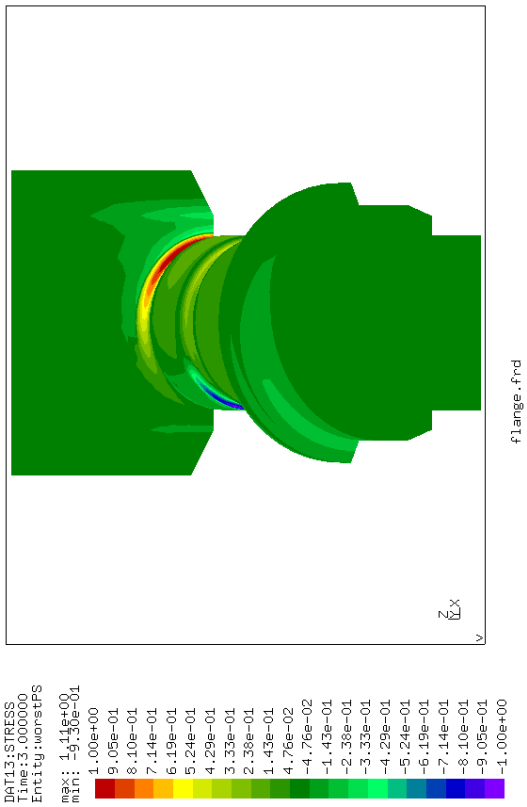


Figure 54c. WorstPS, LGC3.

6.5 Flange 2 (FL2)

In the flange 2 example, two parts are attached together by three bolts. This is shown in the figure 55. The example has multiple contact zones and plastic material behavior is taken into consideration. The mesh consists of quadratic brick and wedge elements. The contact parameters are presented below:

Contact parameters

$K = 10^7$
$\sigma_\infty = 3$
$\mu = 0.1$
$\Lambda = 1*10^5$

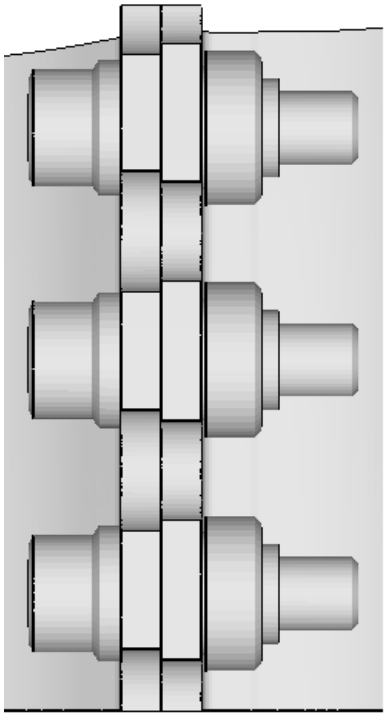


Figure 55. The FL2 –example.

The worst principal stress distribution at the biggest contact zone between the two attached parts is presented in the figures 56a-56c. Nodes version did not converge for this example either so it is omitted. The worst principal stresses given by LGC versions in the figures 56a-56c as well as for the bolts (Appendix VII) are really close to Abaqus. The only significant difference is the stress peaks at the cyclic symmetry cross section. However, this has most likely nothing to do with the used contact formulation. The peak values for LGC versions are also really close to Abaqus.

The pressure distributions are presented in the Appendix VII. The contact zones for LGC versions seem to be slightly larger for the big middle contact area, but actually the pressure distribution for Abaqus is not really smooth either. The maximum values of the pressures for both CalculiX versions are also around 20% lower than the value given by Abaqus.

Figure 56a. WorstPS, LGC7.

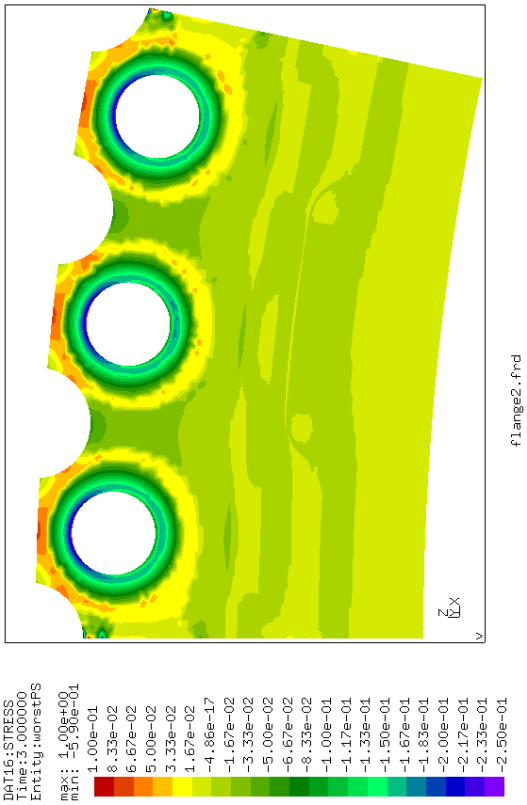


Figure 56b. WorstPS, Abaqus STS.

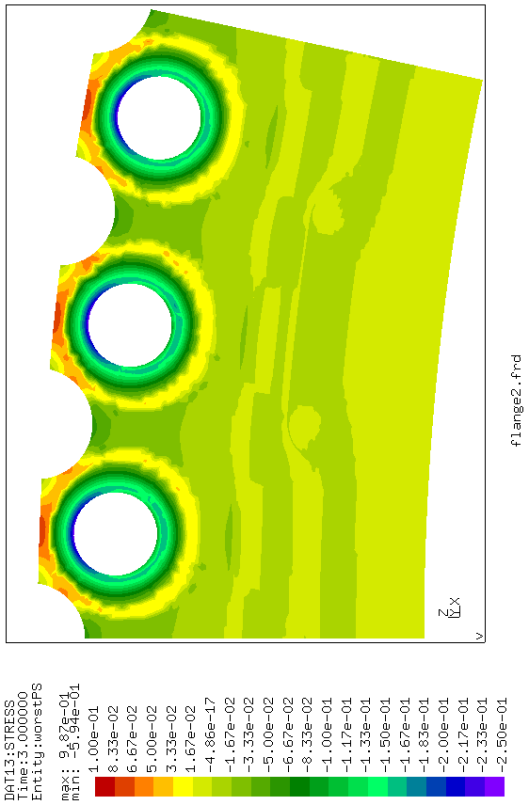


Figure 56c. WorstPS, LGC3.

6.6 Push out test (PO)

In many turbines metal braces called retainers are used to fix turbine blades axially. The retainers are designed to withstand a certain minimum force. The purpose of this test is to verify that the construction of the retainer is sufficient to bear the axial loads. This is highly nonlinear problem with large deformations, which makes the example challenging for finite element analysis. The retainer and the attachment of the foot of the blade are shown in the figures 57 and 58.

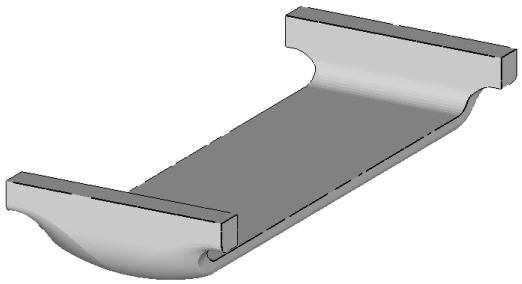


Figure 57. The retainer.

The retainer (figure 57) is placed below the blade foot and an axial force is provided via a rectangular bar. The setup is shown in the figures 59 and 60.

This example was run with two slightly different retainers. First one was the real retainer, which contains confidential geometry and is not presented. This retainer was replaced by another general retainer which contained no confidential geometry (figure 57) to be able to present the results.

The boundary conditions are such that the displacement of the rectangular bar (figures 59 and 60) is predefined. Contact constraints are defined between the rectangular bar and the blade foot, as well as between the blade foot, the retainer and the body. Contact parameters are K = 10^6 and $\sigma_\infty = 3$. Friction was neglected.

The example exhibits large deformations. The resulting displacements are shown in the figures 59 and 60.

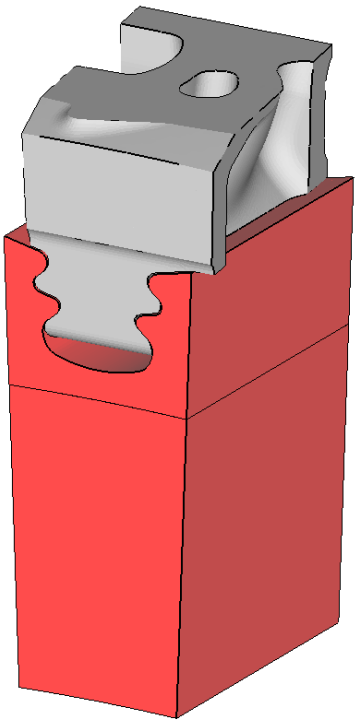
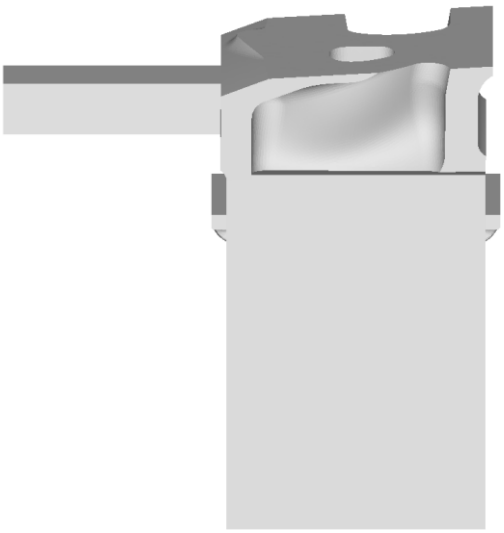


Figure 58. The setup of the push out test example without the retainer.



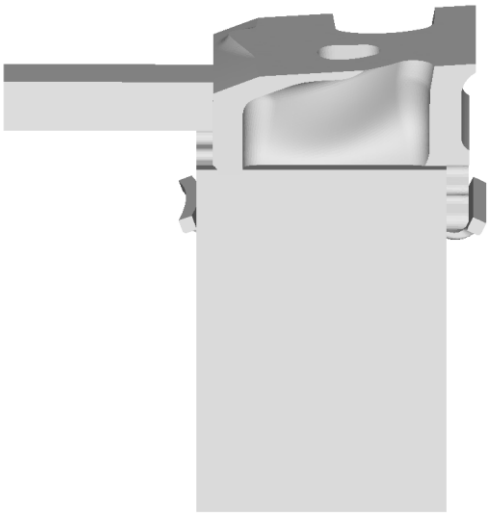


Figure 59. Initial shape of the retainer. **Figure 60. Deformed shape of the retainer.**

Abaqus did not converge for this model, and therefore those results cannot be presented. However, the reason this model was presented, is to illustrate the effect of different increment sizes used by the linear penalty method. This model was run by four different constant increment sizes by LGC7. Increment sizes used were 10%, 25%, 50% and full step. The worst principal stresses for the retainer are presented in the figures 61a-61d. The displacements and the contact pressures for the retainer are presented in the Appendix VIII. For this model, the maximum values are normalized such that the maximum value for increment size of 10% becomes one.

The first thing to note in the worst principal stress distributions is that the maximum values are different. A clear pattern can be recognized; the maximum value seems to grow with increasing increment sizes. For the increment size of 25%, the maximum value is 10% higher compared to the increment size of 10%, for the increment size of 50% it is almost 30% higher and for the increment size of full step, the peak stress is over 40% higher. However, the shapes of the distributions are quite similar.

A similar pattern can be also recognized from the displacements and the pressure distributions. For both of these variables, the maximum values seem to decrease, as the increment size increases. The maximum displacement given by the increment size of full step is about 6% less compared to the increment size of 10%. For the pressures, the peak value is over 50% less. There is also significant difference in the pressure distributions.

After all, the increment size of 10% does not produce perfect results either and nor does the increment size of 1%. It is actually a question about time and accuracy. The computational time for the increment size of 10% is about six times more compared to the increment size of full step. The user must be aware of this disadvantage of the linear version and must decide how much time can be sacrificed to obtain more accurate results.

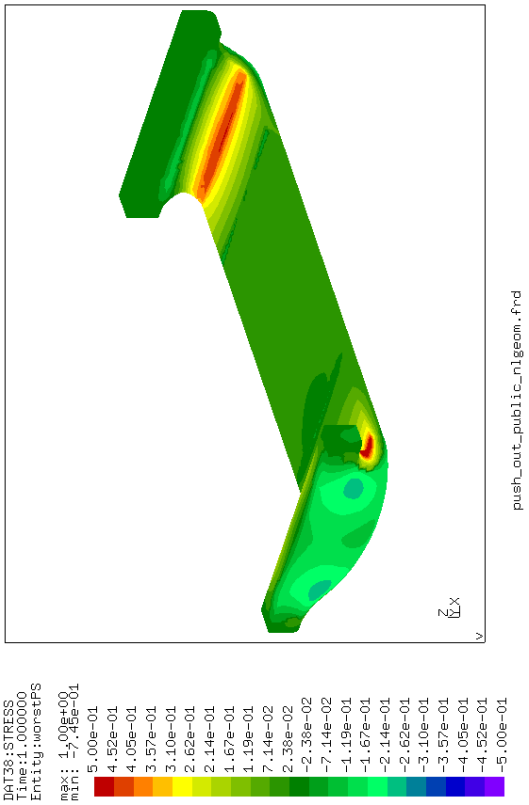


Figure 61a. WorstPS (INC size: 0.1).

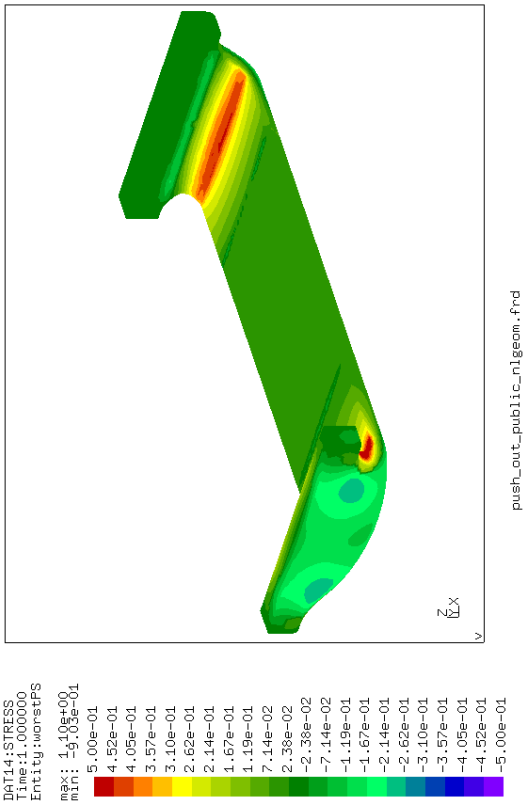


Figure 61b. WorstPS (INC size: 0.25).

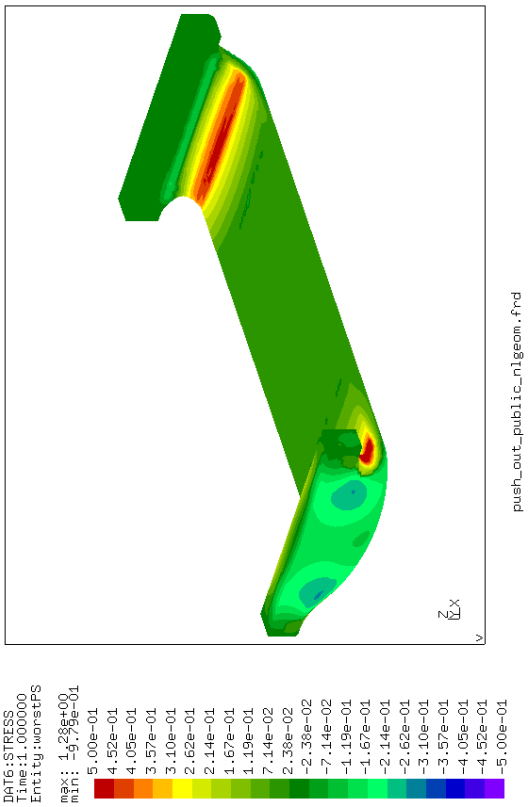


Figure 61c. WorstPS (INC size: 0.5).

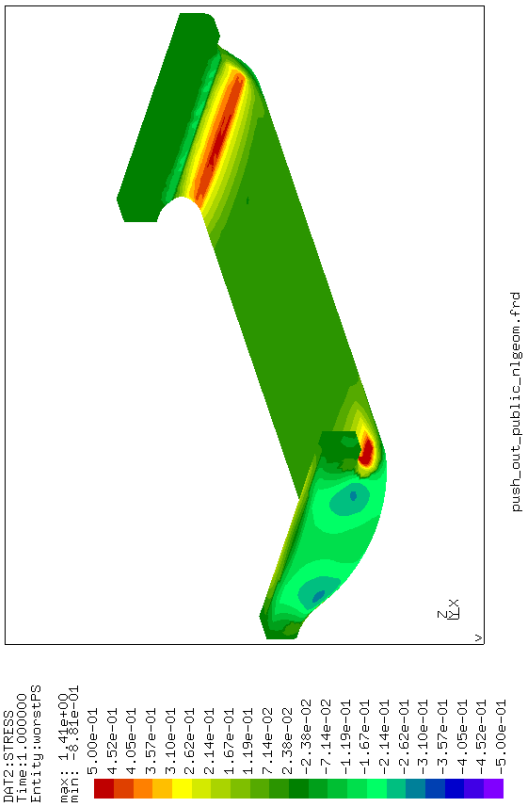


Figure 61d. WorstPS (INC size: 1).

7 Discussion

In this chapter, the theory and the results presented earlier are briefly covered. Advantages of the segment-to-segment formulation as well as the problems related to the linear version are discussed. At the end of the chapter, a conclusion about the master's thesis work is presented.

7.1 Theory

The Gauss version rests on a constant amount of facial integration points, which is consistent with the integration point scheme of the element. The Gauss Cut version uses the triangulation of the contact surfaces to determine the integration point locations. The main difference between these implementations is that for the Gauss version, the local integration points within the face are constant. The one advantage of this formulation is that it is simpler and easier to implement. For the Gauss version, only one facial integration point scheme, which is consistent to the element integration points, was experimented. Further experiments could be conducted for the Gauss version by creating a denser constant integration point scheme. This could lead to better convergence characteristics.

On the other hand, denser integration point schemes were already introduced by the Gauss Cut formulation. The advantage of Gauss Cut version is that the integration point scheme is automatically denser near the borders of the element faces, because the

triangulation is actually based on this cutting of the surfaces (section 4.3). This should reduce the non-physical jumps in the contact forces and lead to a better convergence.

The linear formulation further improves the convergence characteristics. It significantly simplifies the stiffness matrix, which is used to solve the displacements. However, the simplification of the equations comes at a cost. The truly linear pressure-overclosure relationship used in the linear version requires the contact element prerequisites to be checked every iteration, such that no tensional contact forces can appear. Secondly, the normal of the master face is kept constant during the increment. This may lead to false results, if there are significant displacements within the increment. During the increment, the overlapping of the integration point is calculated relative to its point of projection at the beginning of the increment. This location might not be located "underneath" the integration point after few iterations. This is illustrated in the figure 62.

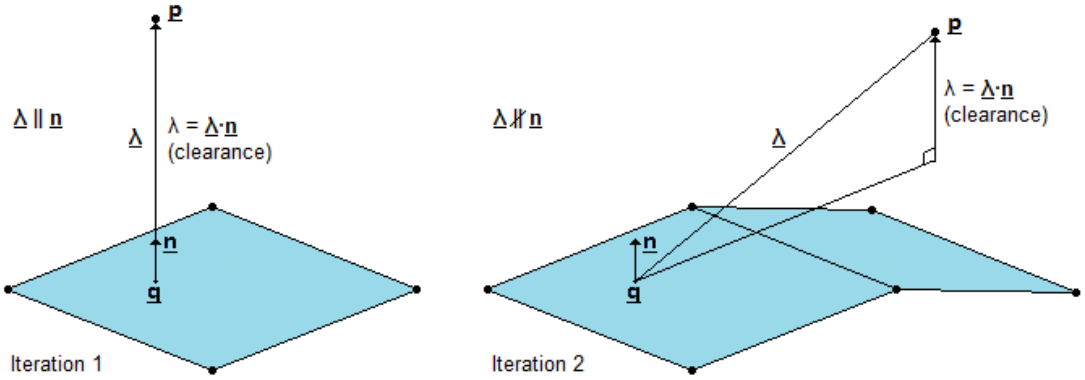


Figure 62. The illustration of the problem caused by a constant normal. Erroneous results may occur due to the displacements after the first iteration.

If the displacements are relatively small, the aforementioned problem can be simply solved by introducing a small additional increment at the end of the step. During this increment, the equilibrium state is calculated by the updated normal vectors of the master face. If there are significant displacements, the increment sizes should be kept small enough throughout the step, such that no erroneous results are calculated at any point.

It would be also possible to introduce an automatic check into the program, which would check if the normal of the master surface is still accurate and can be used to calculate the overlapping. This could be done by comparing the length of the vector $\boldsymbol{\lambda}$ (equation 17) and the overlapping λ (equation 18). Ideally these values should always be identical (figure 62). If the difference would be too large, the program could automatically restart the increment by applying smaller increment size.

The linear version requires a more complicated detection for diverging residuals as was written in the section 4.4. The criteria regarding the determination of divergence were

developed mainly by studying the convergence behavior of the small amount of real models related to turbine engines. The divergence to be detected requires the one of the following three occurrences to happen (section 4.4.3): the mean residual force has increased too much, the amount of contact elements is stabilized, or a repetitive pattern in the residual force can be identified from consecutive iterations. The value which determines whether the mean residual force has grown too much to allow divergence is chosen arbitrarily based on experiments. Similarly, the determination of repetitive pattern is based on experiments. No comprehensive comparison has been conducted for these requirements, which means that it is highly possible to achieve better outcome through extensive experimentation and optimization. Furthermore, at the time of the writing only the repetitive patterns in which the repetitive behavior occurs up to every two consecutive iterations are detected. However, it is noticed that the repetition may also require more than two iterations to recur, even though this behavior is quite rare. Currently, the program does not detect this kind of behavior and just iterates until the maximum number of iterations within an increment is reached.

Modeling of tied contact was also experimented by slightly modified LGC7 segment-to-segment formulation. The truly linear pressure-overclosure relationship of the linear version is already able to provide tensional contact forces needed by the tied contact calculation. First of all, c_0 was set to a very high number such that contact elements are always created if an opposite master surface is present. Secondly, the friction coefficient was also set to a very high number and the tangent of the stick range was set equal to the spring coefficient. This arrangement made sure that the displacements to all directions are equally constrained and no tangential slip occurs. As a result, tied contact can be accurately modeled by the linear segment-to-segment formulation. Implementation of this segment-to-segment tied contact formulation into the program would not be a big task since the modifications required are quite small.

7.2 Results

At first, the nonlinear versions of segment-to-segment penalty formulations were introduced. The convergence characteristics improved noticeably when the stability of the simple cube and the punch examples were compared. However, the old problem in convergence with quadratic elements existed and the convergence improvement of the many real examples was not really significant.

The introduction of the linearized version made actually a huge further improvement for the stability of the segment-to-segment method. For the linear version, quadratic elements cause no problems, because the point of projection onto the master surface as well as the normal at this point is constant. Even though it is known that the linear version suffers from the possible accuracy problem, the results have been quite accurate anyway. The most examples however did not exhibit large displacements. It actually seems that for examples in which there are no large displacements, good results can be achieved even with big increment sizes. However, it is important that the user realizes that the accuracy decreases with increasing increment sizes.

No significant differences in most results were noticed between Gauss and Gauss Cut versions. Furthermore, one, three, seven, and twelve point schemes used in Linear Gauss Cut version gave quite similar results as well. The convergence of the real examples was also more or less similar between all linear versions. However, the simple test examples (chapter 5) with coarser mesh weakly indicate that the convergence actually improves with increasing amount of integration points. This is probably the correct conclusion, but the mesh used in real models is usually relatively dense such that even the one-point scheme seems to be enough in most cases.

The patch test for normal contact, but also for tied contact was conducted. It resolves whether the correct stress state can be transferred across the contact interface of a given pair of elastic contacting bodies. The results from the patch tests clearly demonstrate the one significant disadvantage of node-to-segment formulations. The results for Abaqus NTS, Nodes and Linear Gauss were really bad. Even though Linear Gauss version has performed well for many other examples, it does not pass the patch test. This originates from the fact that the integration point scheme of the Gauss formulation is based only on the mesh of the slave surface. If this mesh is coarse, there are simply not enough integration points to transfer the correct stress state to the master side. This is not the case with the Gauss Cut formulations since the integration points are determined based on the cutting of the slave and the master surfaces hence the integration points do not depend on the choice of the slave surface.

Abaqus STS and LGC formulations performed much better for the patch test. Abaqus STS gave the worst results of these formulations, but the LGC1 formulation performed only slightly better. The results from other LGC versions with denser integrations point schemes were clearly better and almost ideal. Similar results were obtained for the tied contact patch test. The present tied contact formulation which is based on MPC's performed similarly to the node-to-segment contact formulations. LGC7 formulation performed again almost ideally.

In general, the results from LGC versions for small and simple examples as well as big real models were good. The stress and the displacement distributions, but also the maximum values of these variables were really close to Abaqus STS. The distributions of the contact variables were also similar, but differences in the maximum values were noticed. However, for most of the examples, these maximum values are still better compared to the node-to-segment penalty version.

7.3 Conclusion

This thesis introduced a transition from the node-to-segment penalty method to the segment-to-segment penalty method. The new formulation was implemented to the finite element analysis program CalculiX. The main issue of the node-to-segment contact formulations is the discontinuities in the contact forces. These discontinuities are encountered when individual nodes slide between adjacent master faces or slide off

the boundaries of the master surface. Additional drawback of the node-to-segment formulations is a significant loss in accuracy in case of nonconforming meshes. The proposed segment-to-segment approach deals with all of these issues and leads to a better stability and accuracy.

Other remarkable subject is the linear contact formulation. The linear version does not update the point of projection of the integration point onto the master surface every iteration, but only once per increment. Additionally, the linear version applies truly linear pressure-overclosure relationship instead of quasi bilinear. These properties of the linear version simplify the resulting stiffness contributions and yield a much better convergence. The disadvantages of the linear version were acknowledged and discussed. The most remarkable drawback is that the accuracy of the linear version significantly reduces with increasing increment sizes. This applies especially for examples which exhibit large deformations. It is important that the user is aware of this phenomenon.

The new contact formulations with different amount of integration points were compared by small, simple test examples, but also with big real models. The better stability of the segment-to-segment formulation was discovered immediately with small examples. Even better convergence and stability was achieved by introducing a linear version. The linear version has good convergence characteristics even with quadratic contact surfaces which made re-meshing of the contact surfaces pointless, and it was removed from the program. In general, the results are good, even though the used increment sizes may have affect to the results. Moreover, the patch test demonstrated the overwhelming accuracy of LGC formulations.

Summarizing the results, the Linear Gauss Cut version with seven integration point scheme (LGC7) was chosen to be implemented for the next official CalculiX version. It replaced the earlier mortar method and was implemented as an alternative contact method opposed to the node-to-segment penalty method. Even though all linear versions performed quite well with real models, the small examples gave an indication of the fact that less than seven integration points per triangle might not be enough in all occasions. Moreover, there were no noticeable difference in terms of iterations, and the increase in computational time per one iteration is quite small. After all, the desired contact formulation was successfully developed with good results.

References

Annavarapu, C. 2014, "A Nitsche stabilized finite element method for frictional sliding on embedded interfaces. Part I: Single interface", *Computer Methods in Applied Mechanics & Engineering,* , pp. 417-436.

Baig, M. 2006, *A Consistent Segment Procedure for Solution of 2D Contact Problems with Large Displacements*, Massachusetts Institute of Technology.

Chouly, F. & Hild, P. 2013, "A Nitsche-based method for unilateral contact problems: numerical analysis", *SIAM Journal on Numerical Analysis,* vol. 51, no. 2, pp. 1295-1307.

Chouly, F., Hild, P. & Renard, Y. 2013, "Symmetric and non-symmetric variants of Nitsche's method for contact problems in elasticity: theory and numerical experiments".

Dhondt, G. 2004, *The finite element method for three-dimensional thermomechanical applications,* Wiley, Chichester.

Dhondt, G. 2013, "CalculiX CrunchiX USER'S MANUAL version 2.6.1", http://www.dhondt.de/ccx_2.6.1.pdf.

Dhondt, G. [no date], "CalculiX: A Three-Dimensional Structural FiniteElement Program" http://www.dhondt.de/.

Gu, R., Murty, P. & Zheng, Q. 2002, "Use of penalty variable in finite element analysis of contacting objects", *Computers & Structures,* vol. 80, no. 31, pp. 2449-2459.

Hallquist, J., Goudreau G. & Benson D., 1985, "Sliding interfaces with contact-impact in large-scale Lagrangian computations", Computer Methods in Applied Mechanics and Engineering, Vol.51(1), pp.107-137, vol. 51, no. 1, pp. 107-137.

Jahn, S. 2007, "Qucs, Technical papers", http://qucs.sourceforge.net/tech/node29.html.

Miller, S.J. 2006, "The Method of Least Squares", *Mathematics Department Brown University, Providence, RI,* .

Puso, M.A. & Laursen, T.A. 2004, "A mortar segment-to-segment frictional contact method for large deformations", *Computer Methods in Applied Mechanics and Engineering,* vol. 193, no. 45, pp. 4891-4913.

Sloan, S.W. 1987, "A fast algorithm for constructing Delaunay triangulations in the plane", *Advances in Engineering Software (1978),* vol. 9, no. 1, pp. 34-55.

Taylor, R.L. & Papadopoulos, P. 1991, "On a patch test for contact problems in two dimensions", *Computational methods in nonlinear mechanics,* , pp. 690-702.

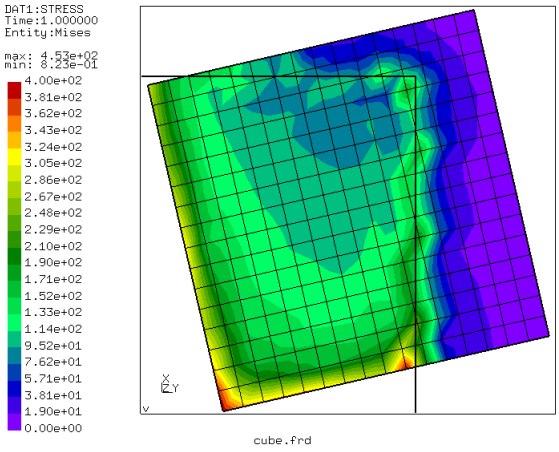
Weyler, R., Oliver, J., Sain, T. & Cante, J.C. 2012, "On the contact domain method: A comparison of penalty and Lagrange multiplier implementations", *Computer Methods in Applied Mechanics and Engineering,* vol. 205–208, no. 0, pp. 68-82.

Wittig, K. 2013, "CalculiX USER'S MANUAL - CalculiX GraphiX, version 2.6", http://www.dhondt.de/cgx_2.6.1.pdf.

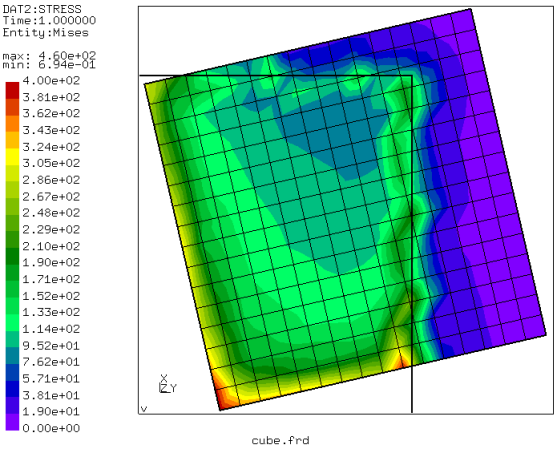
Zavarise, G. 1999, "A superlinear convergent augmented Lagrangian procedure for contact problems", *Engineering Computations, 1999, Vol.16(1), pp.88-119,* vol. 16, no. 1, pp. 88-119.

Zavarise, G. & De Lorenzis, L. 2009, "The node-to-segment algorithm for 2D frictionless contact: Classical formulation and special cases", *Computer Methods in Applied Mechanics and Engineering,* vol. 198, no. 41–44, pp. 3428-3451.

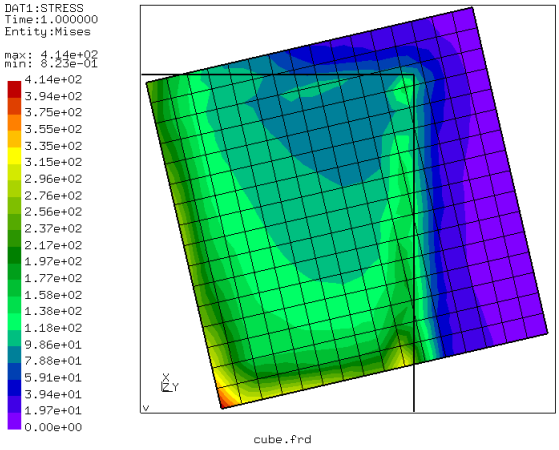
Appendix I: Cube



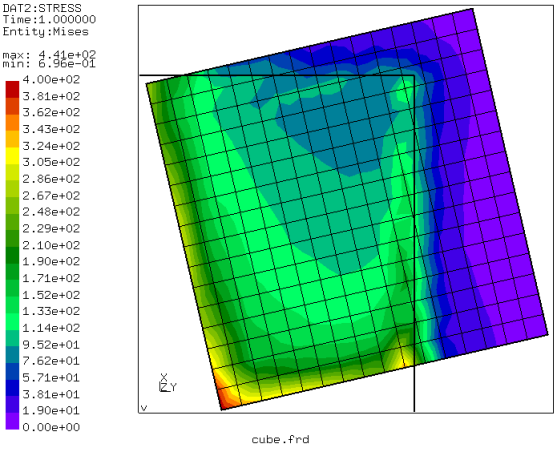
Von Mises stress, Abaqus NTS.



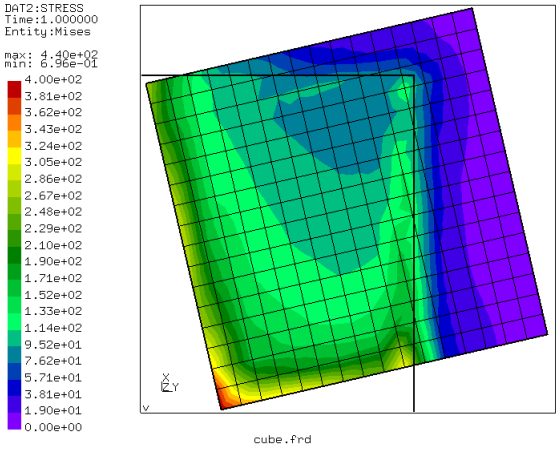
Von Mises stress, Nodes.



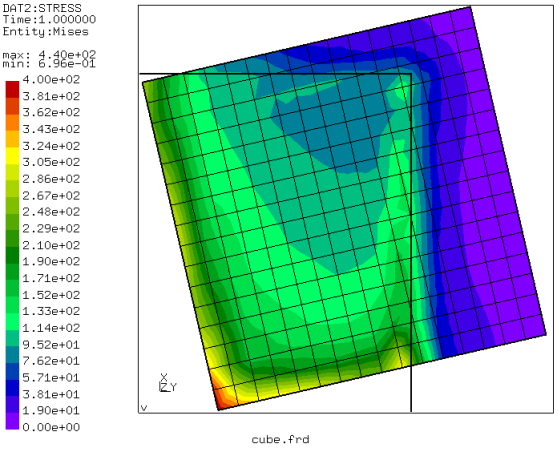
Von Mises stress, Abaqus STS.



Von Mises stress, Linear Gauss.

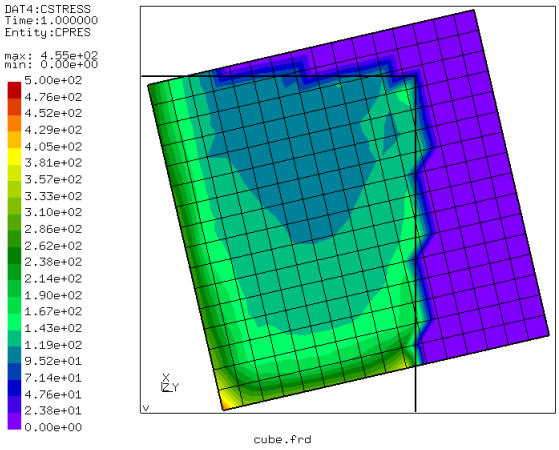


Von Mises stress, GC7.

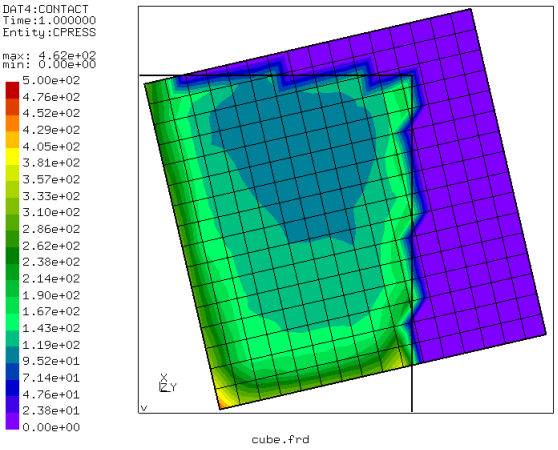


Von Mises stress, LGC7.

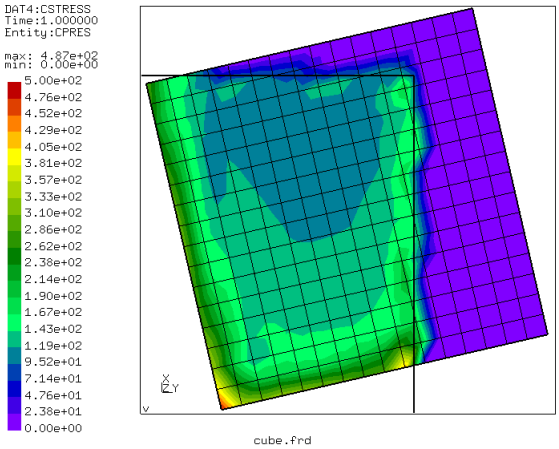
Appendix I: Cube



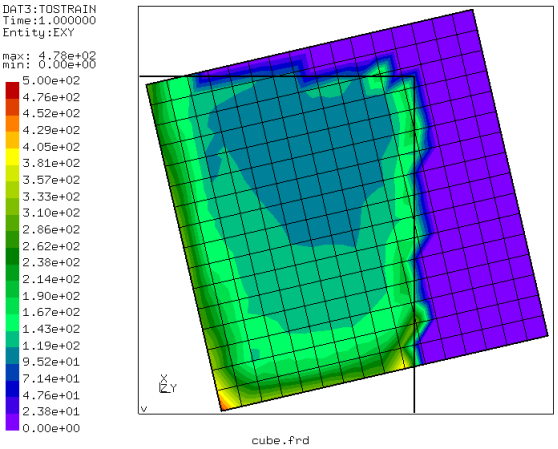
Contact pressure, Abaqus NTS.



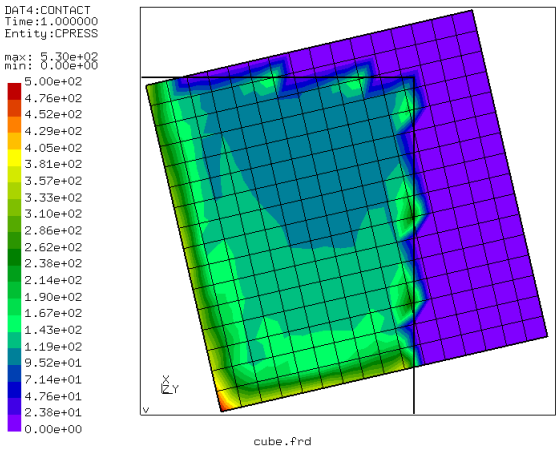
Contact pressure, Nodes.



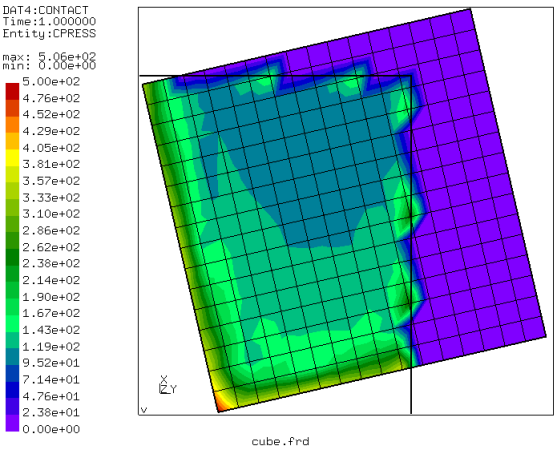
Contact pressure, Abaqus STS.



Contact pressure, Linear Gauss.

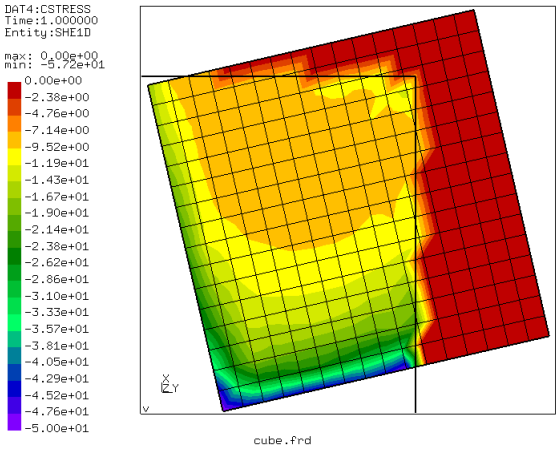


Contact pressure, GC7.

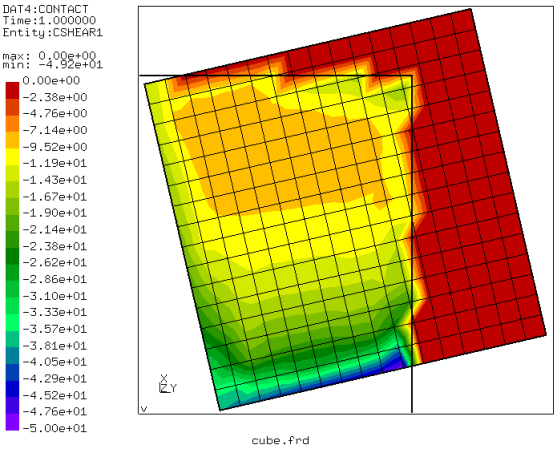


Contact pressure, LGC7.

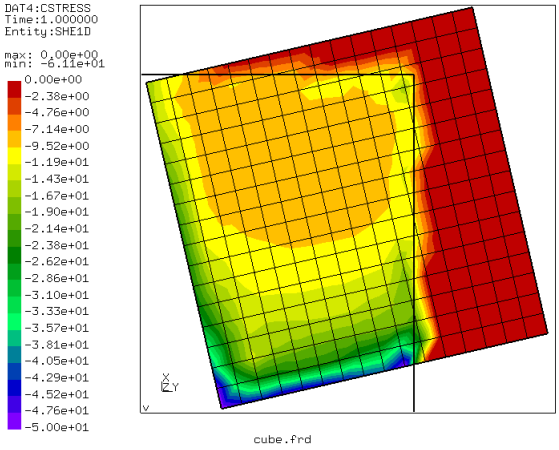
Appendix I: Cube



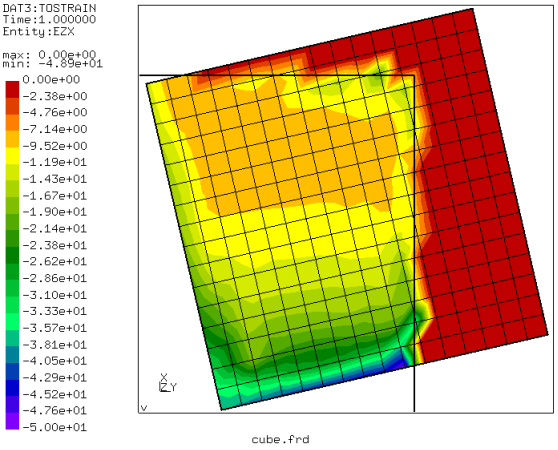
Shear stress (1), Abaqus NTS.



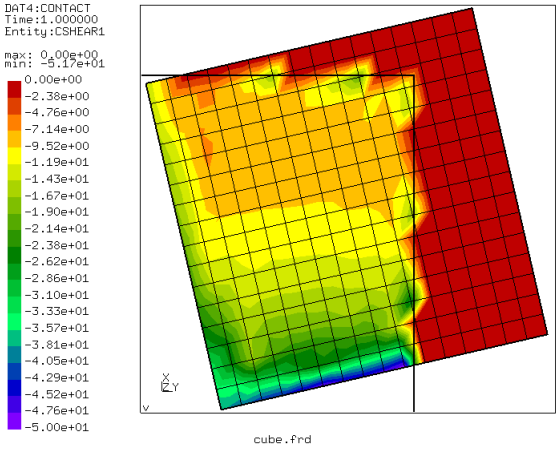
Shear stress (1), Nodes.



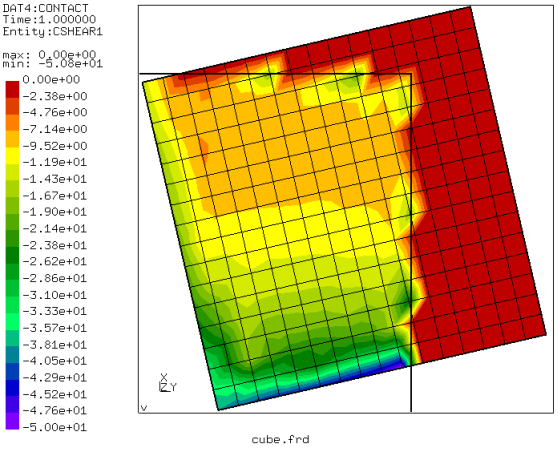
Shear stress (1), Abaqus STS.



Shear stress (1), Linear Gauss.

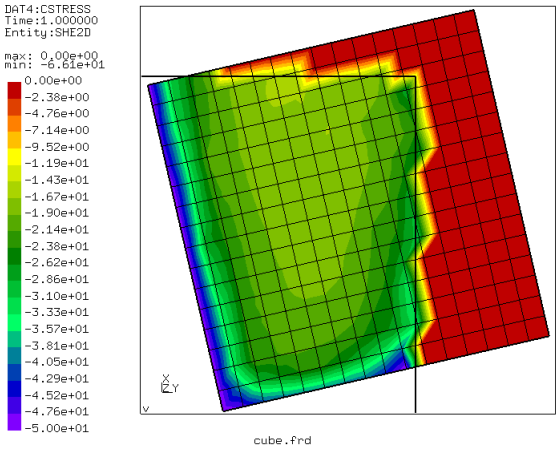


Shear stress (1), GC7.

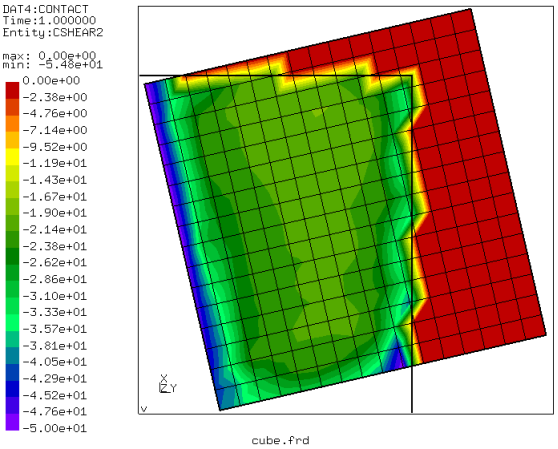


Shear stress (1), LGC7.

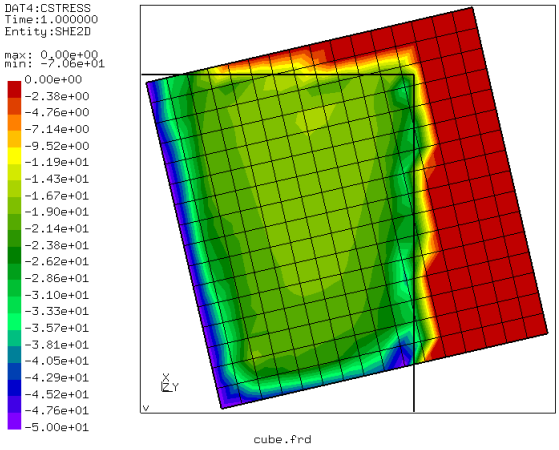
Appendix I: Cube



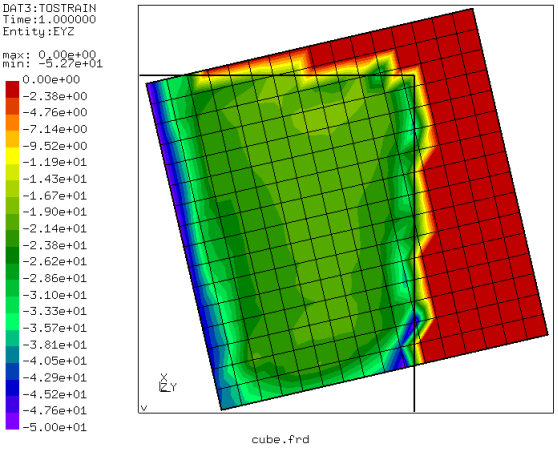
Shear stress (2), Abaqus NTS.



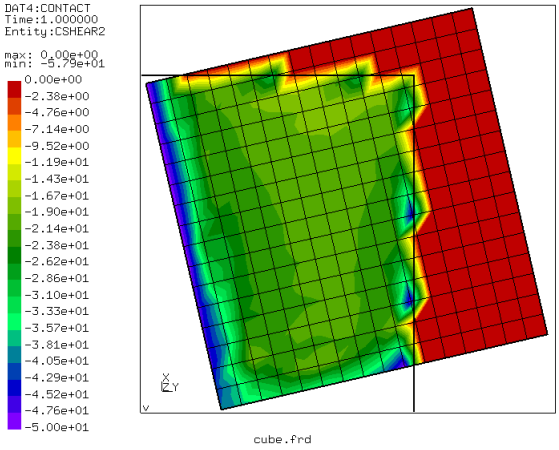
Shear stress (2), Nodes.



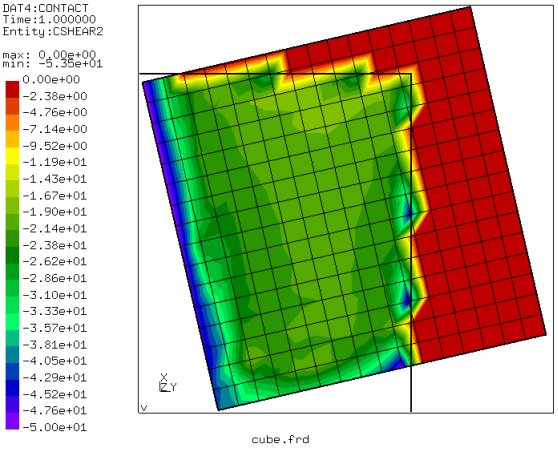
Shear stress (2), Abaqus STS.



Shear stress (2), Linear Gauss.

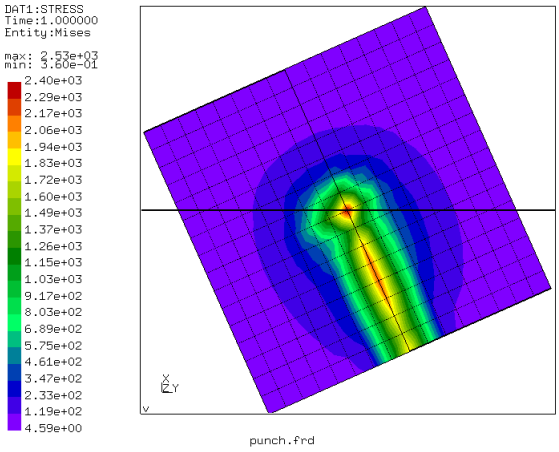


Shear stress (2), GC7.

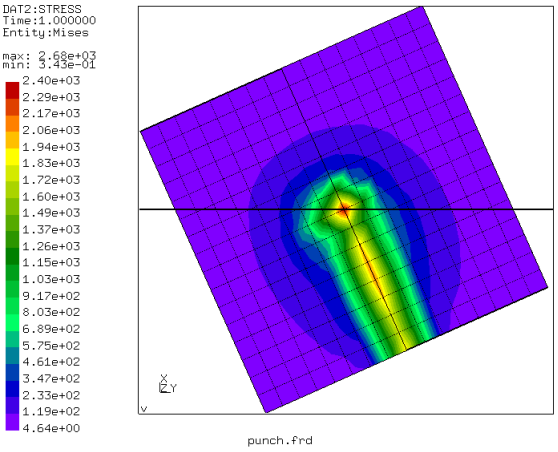


Shear stress (2), LGC7.

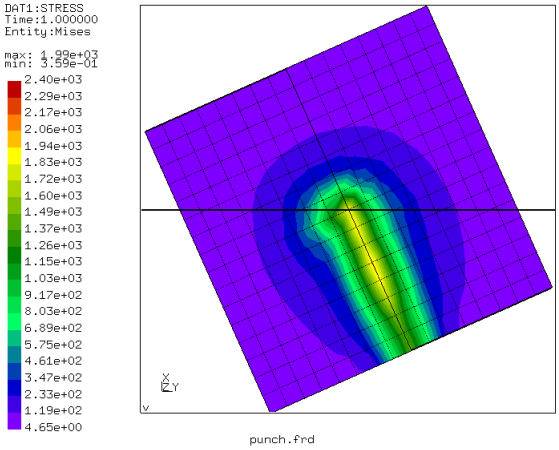
Appendix II: Punch



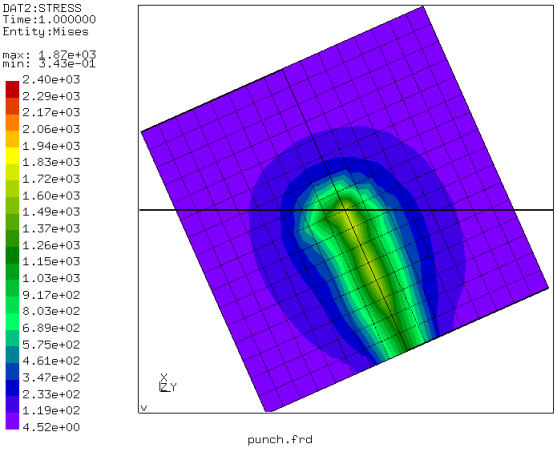
Von Mises stress, Abaqus NTS.



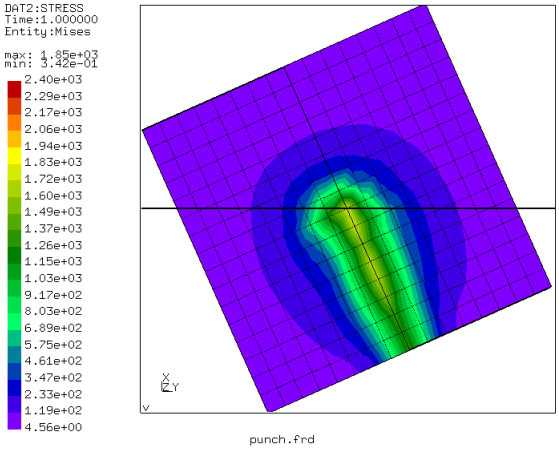
Von Mises stress, Nodes.



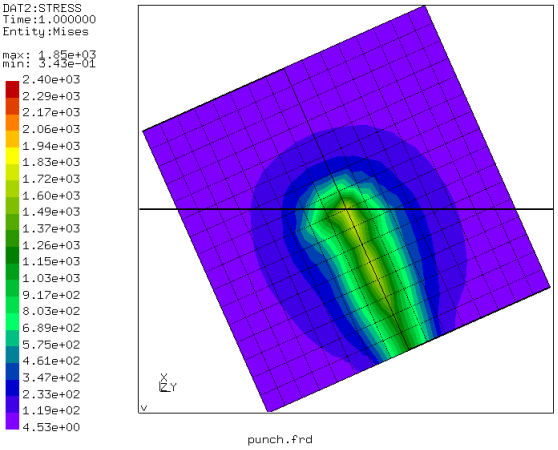
Von Mises stress, Abaqus STS.



Von Mises stress, Linear Gauss.

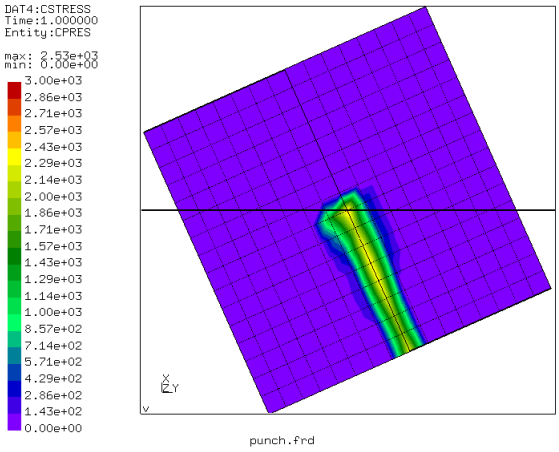


Von Mises stress, GC7.

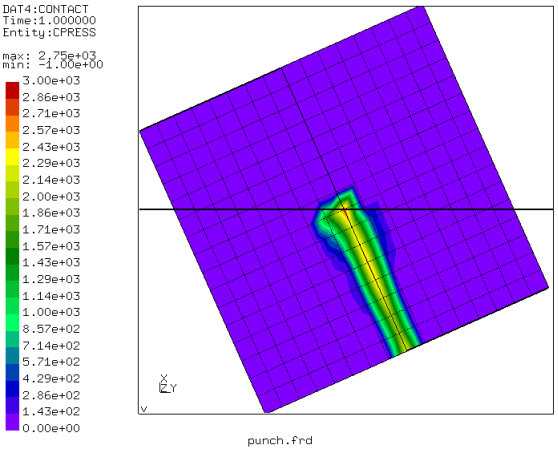


Von Mises stress, LGC7.

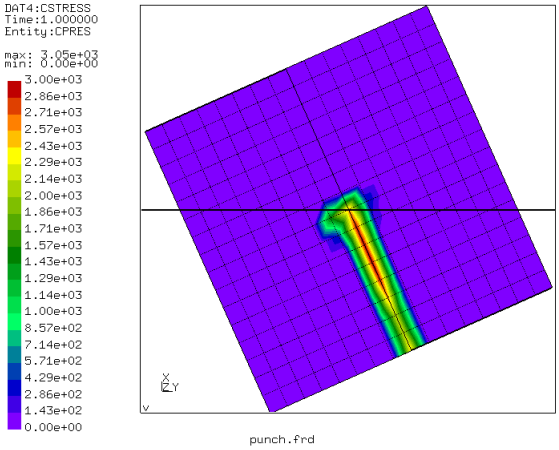
Appendix II: Punch



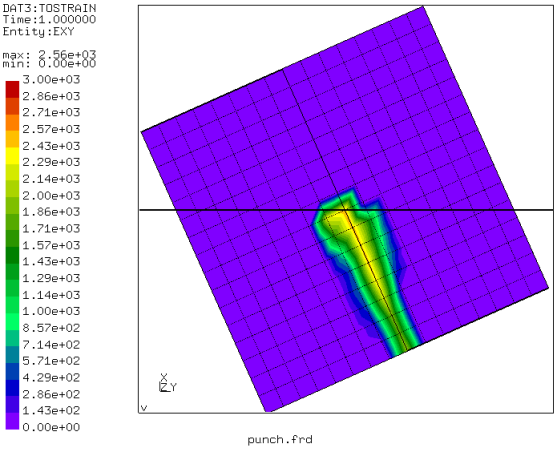
Contact Pressure, Abaqus NTS.



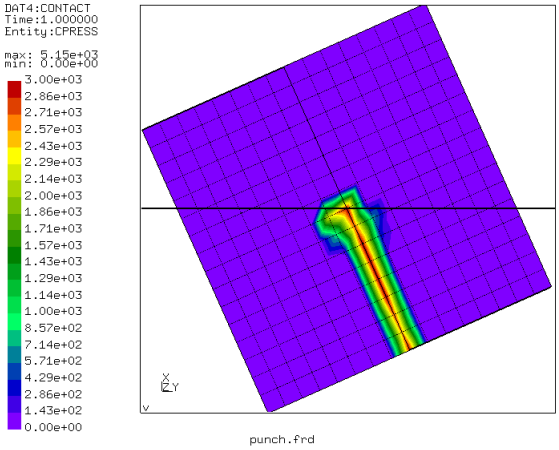
Contact Pressure, Nodes.



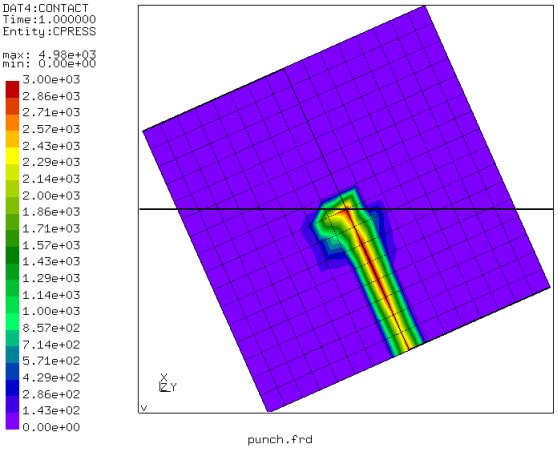
Contact Pressure, Abaqus STS.



Contact Pressure, Linear Gauss.

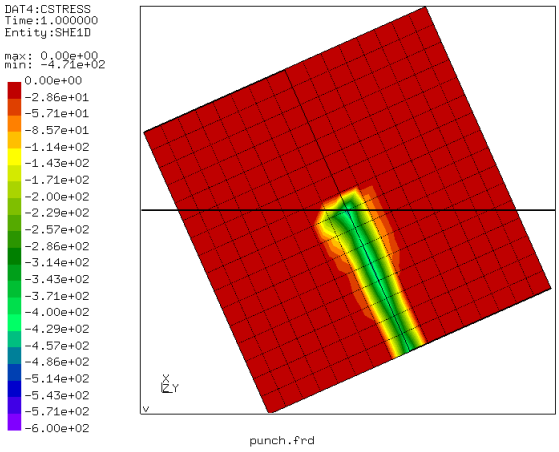


Contact Pressure, GC7.

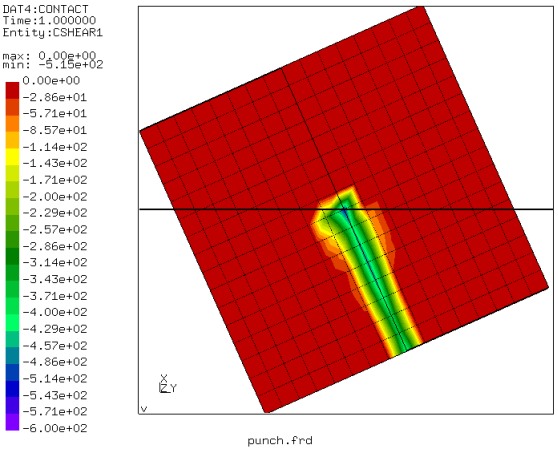


Contact Pressure, LGC7.

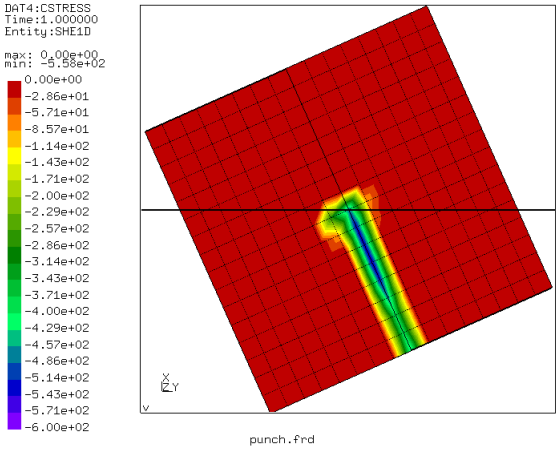
Appendix II: Punch



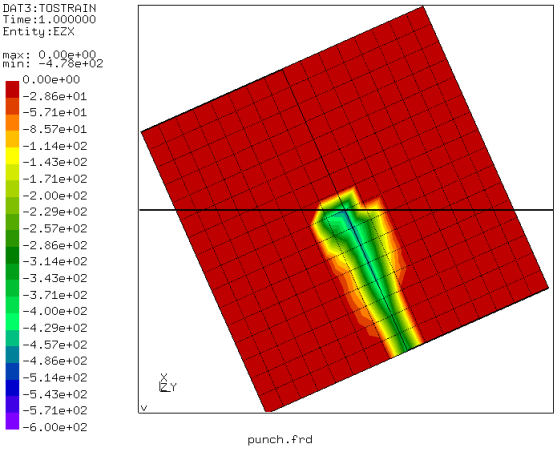
Shear stress (1), Abaqus NTS.



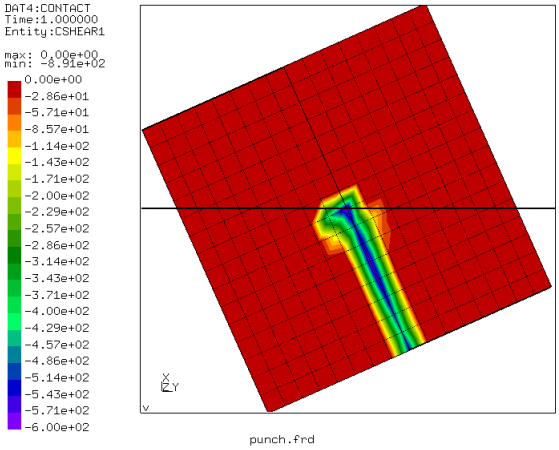
Shear stress (1), Nodes.



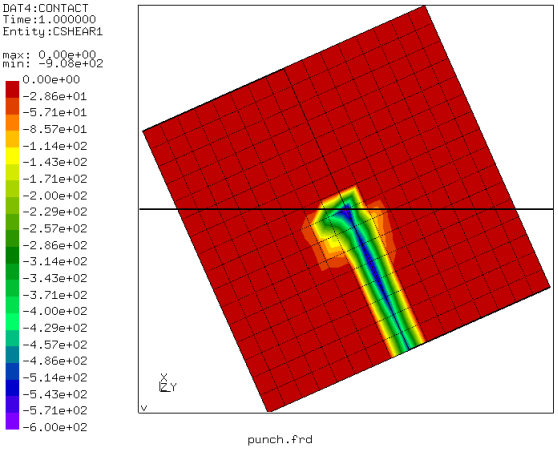
Shear stress (1), Abaqus STS.



Shear stress (1), Linear Gauss.

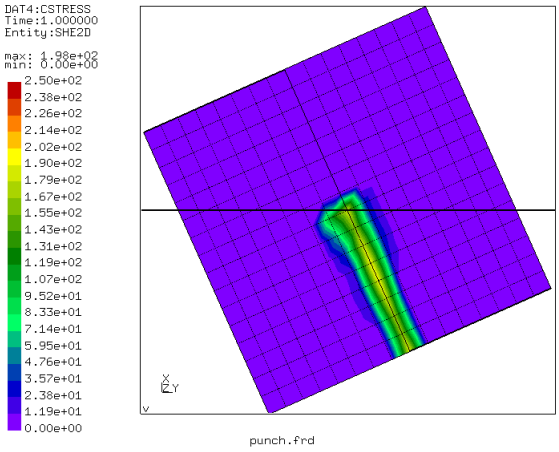


Shear stress (1), GC7.

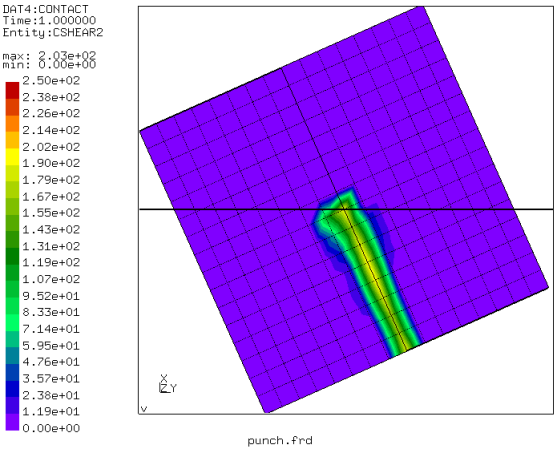


Shear stress (1), LGC7.

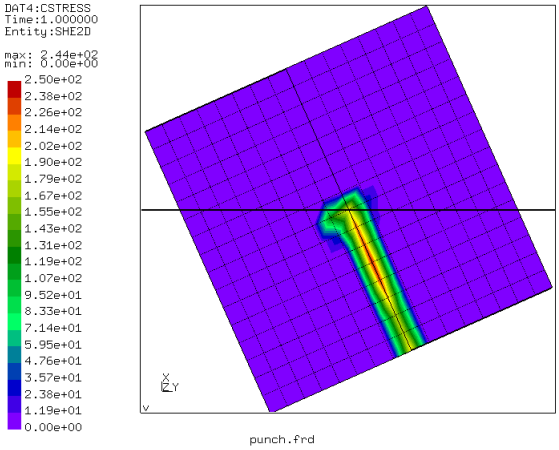
Appendix II: Punch



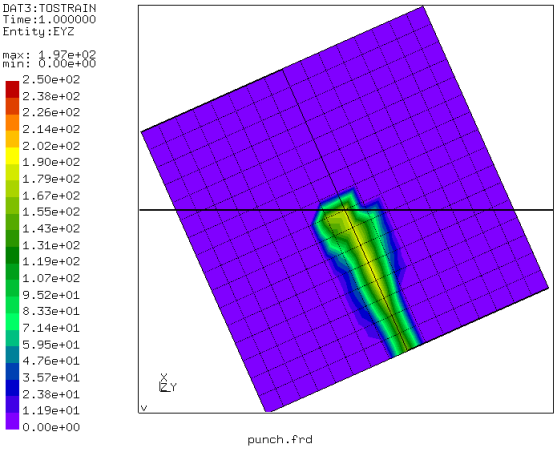
Shear stress (2), Abaqus NTS.



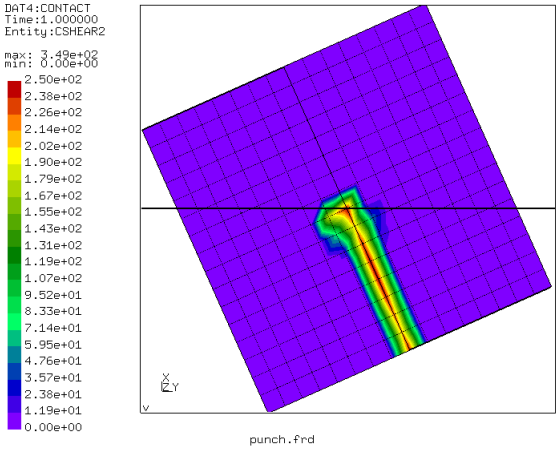
Shear stress (2), Nodes.



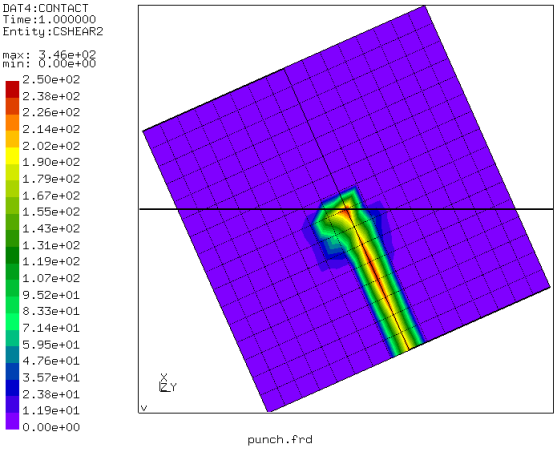
Shear stress (2), Abaqus STS.



Shear stress (2), Linear Gauss.

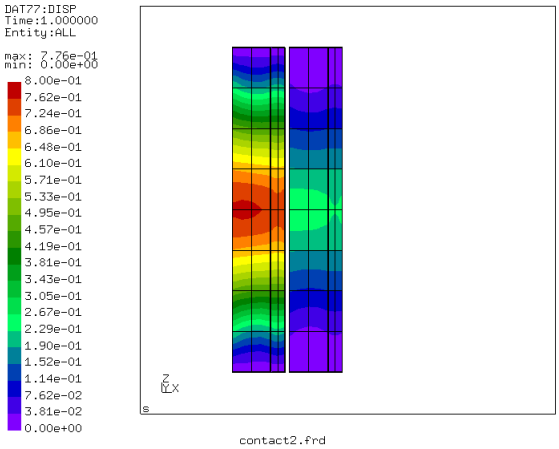


Shear stress (2), GC7.

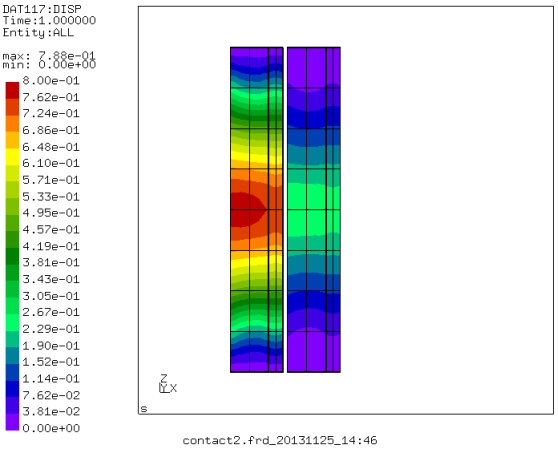


Shear stress (2), LGC7.

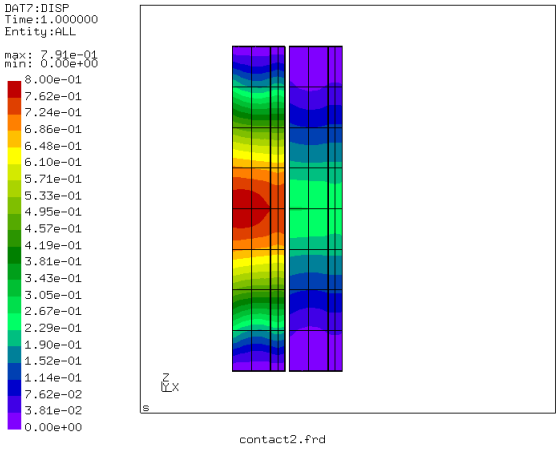
Appendix III: Two beams (1)



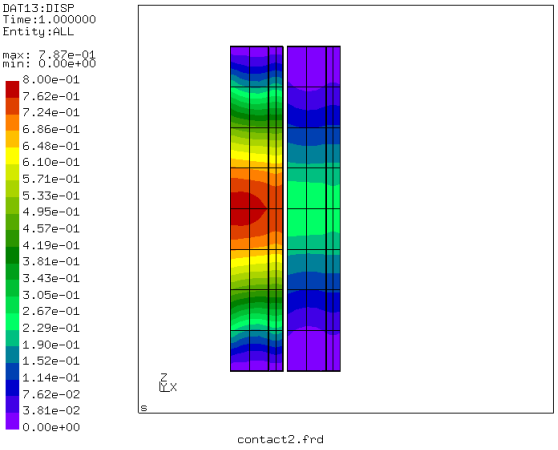
Displacement, Abaqus NTS.



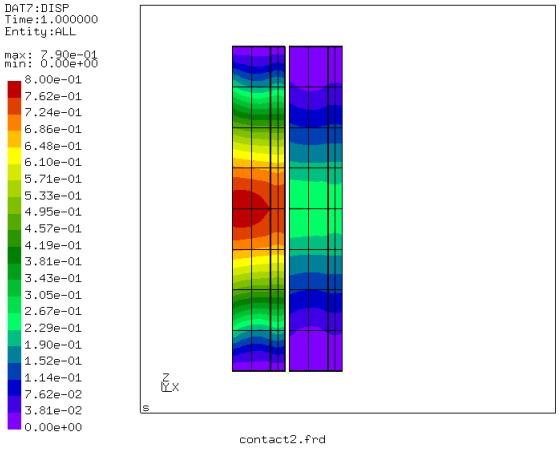
Displacement, Abaqus STS.



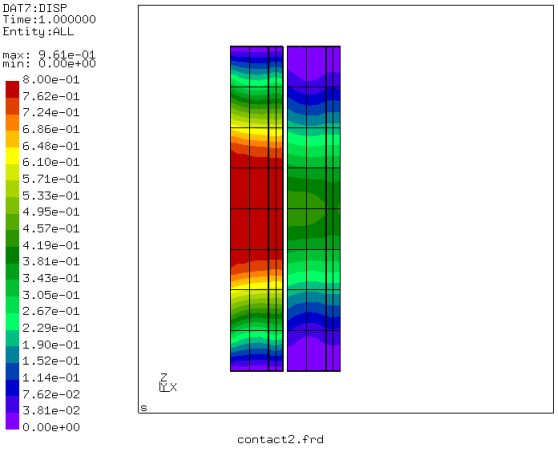
Displacement, LGC3.



Displacement, LGC7.

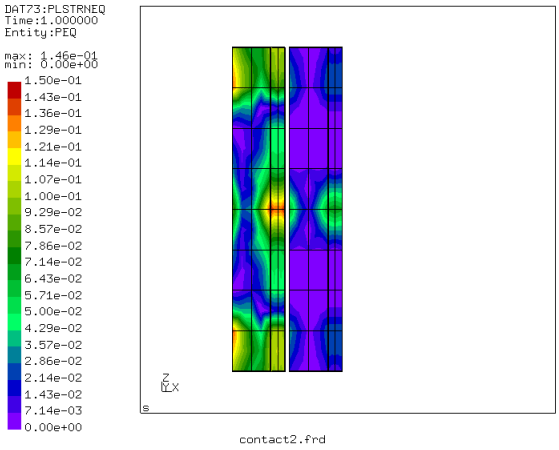


Displacement, LGC12.

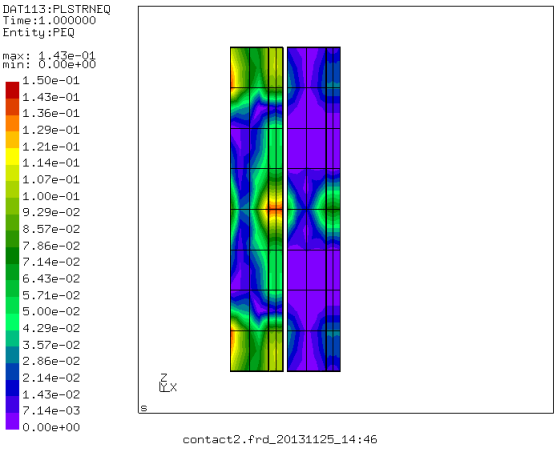


Displacement, Linear Gauss.

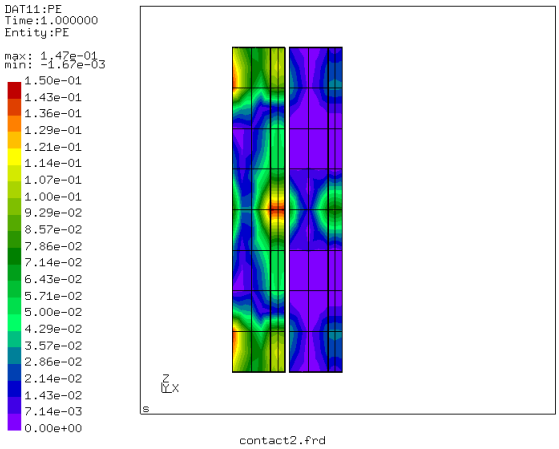
Appendix III: Two beams (1)



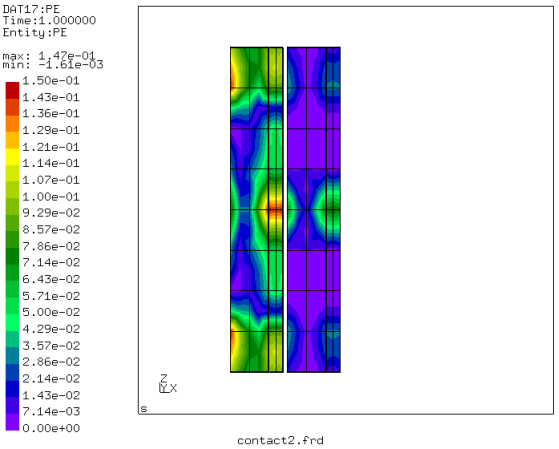
Equivalent plastic strain, Abaqus NTS.



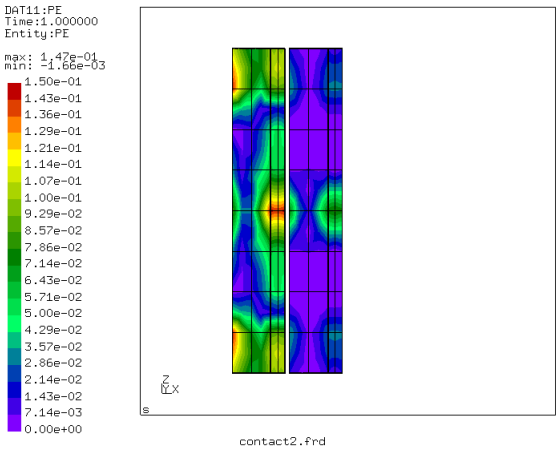
Equivalent plastic strain, Abaqus STS.



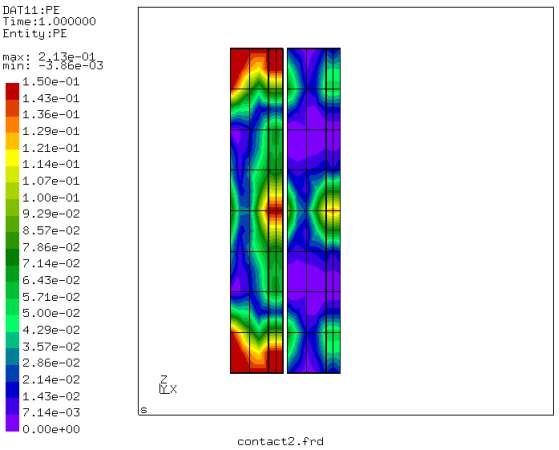
Equivalent plastic strain, LGC3.



Equivalent plastic strain, LGC7.

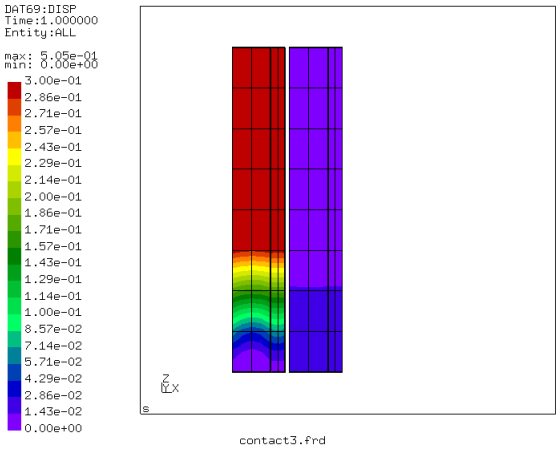


Equivalent plastic strain, LGC12.

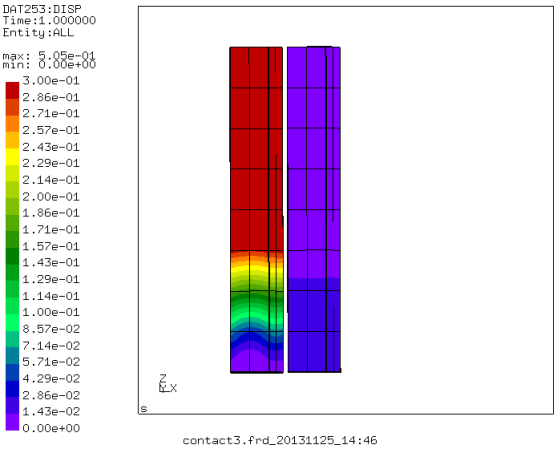


Equivalent plastic strain, Linear Gauss.

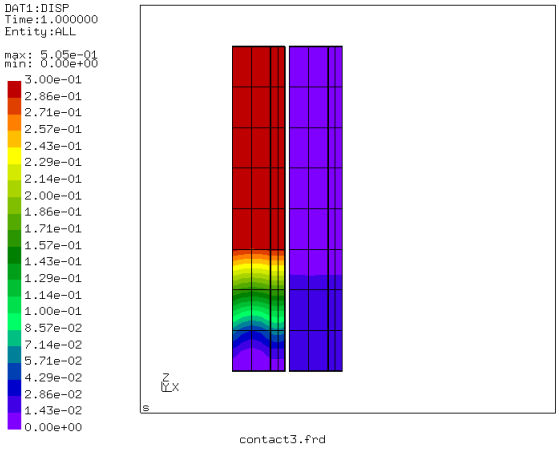
Appendix IV: Two beams (2)



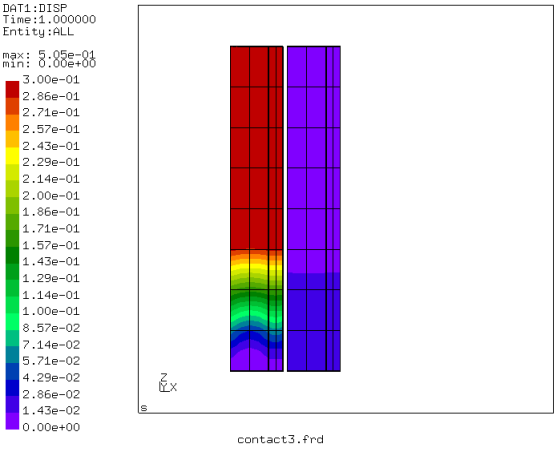
Displacement, Abaqus NTS.



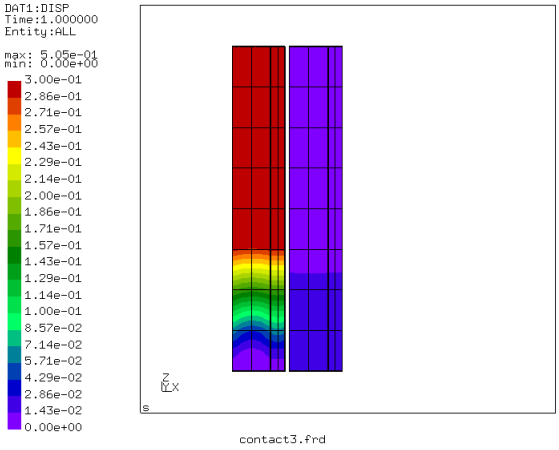
Displacement, Abaqus STS.



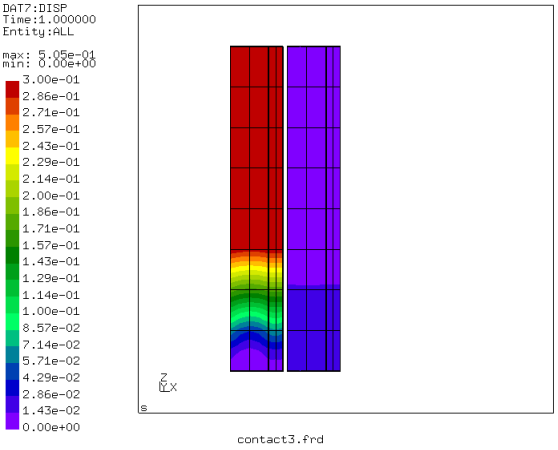
Displacement, LGC3.



Displacement, LGC7.

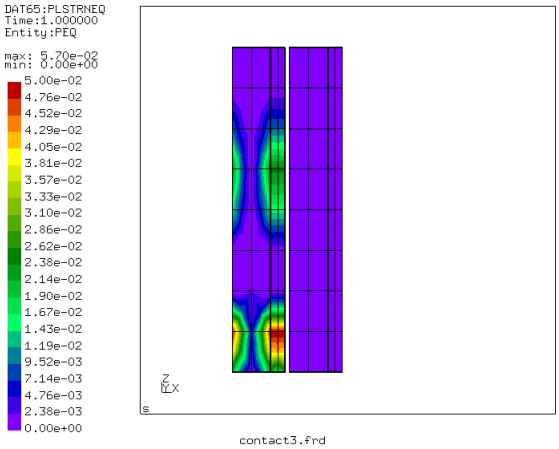


Displacement, LGC12.

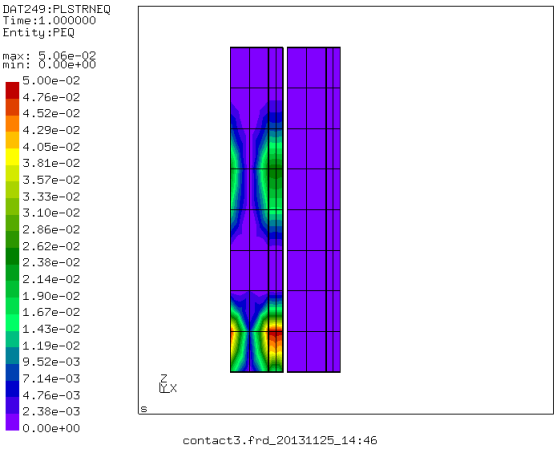


Displacement, Linear Gauss.

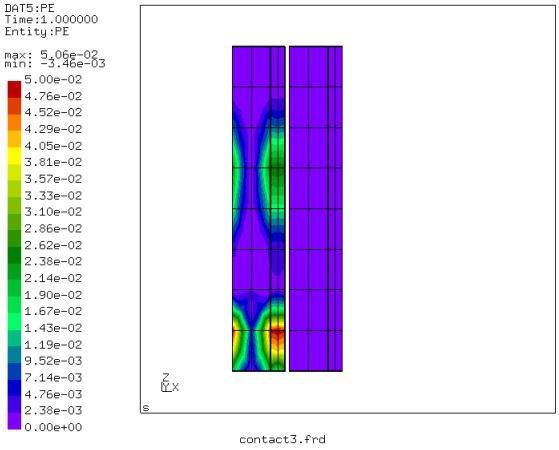
Appendix IV: Two beams (2)



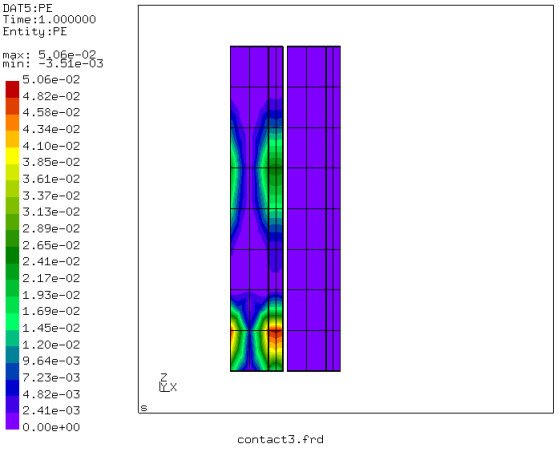
Equivalent plastic strain, Abaqus NTS.



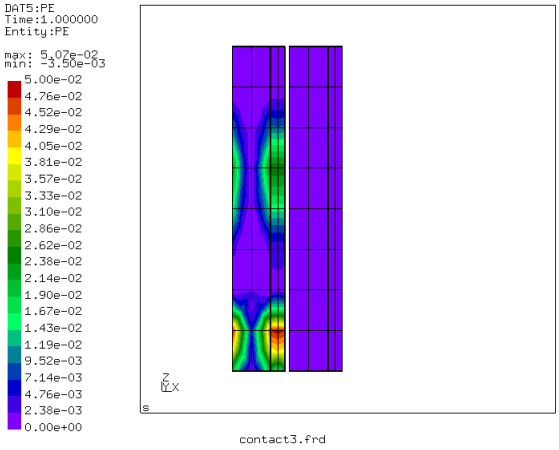
Equivalent plastic strain, Abaqus STS.



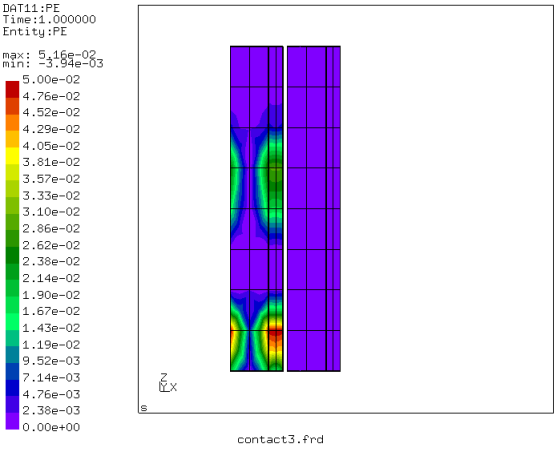
Equivalent plastic strain, LGC3.



Equivalent plastic strain, LGC7.

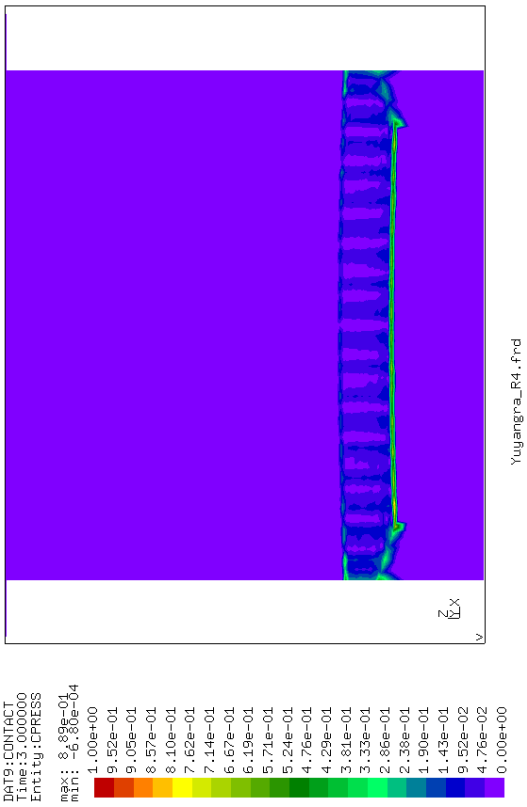


Equivalent plastic strain, LGC12.

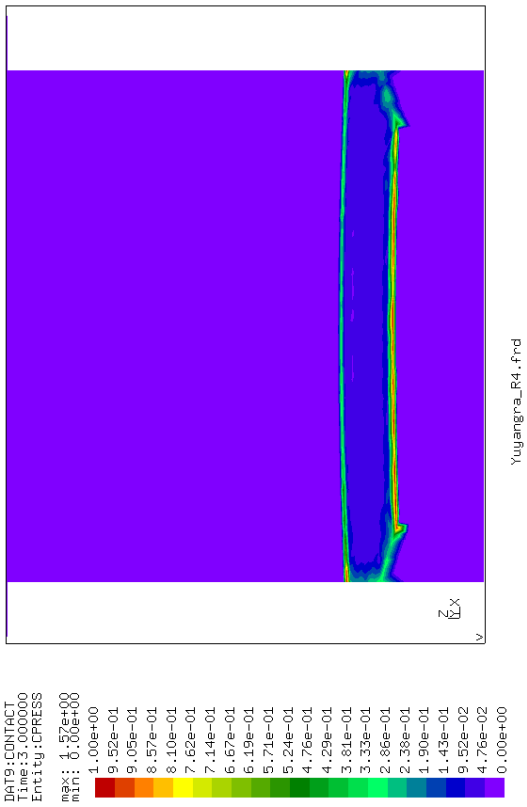


Equivalent plastic strain, Linear Gauss.

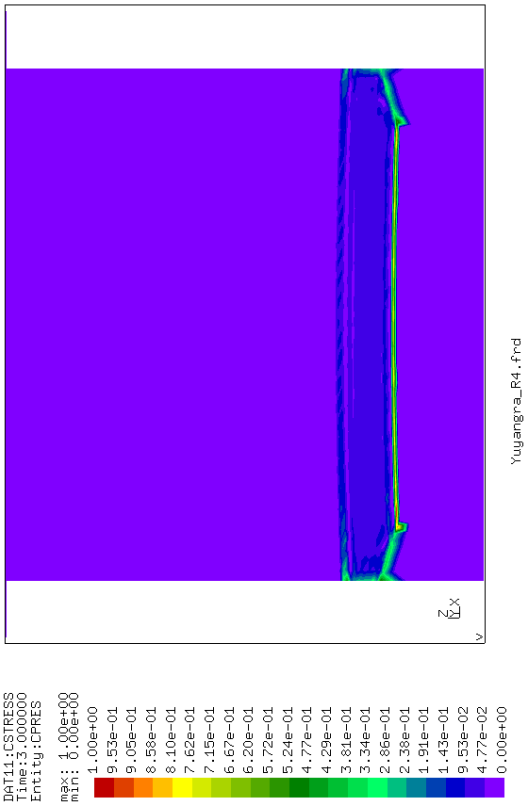
Appendix V: Disk – Blade foot 1 (Y4)



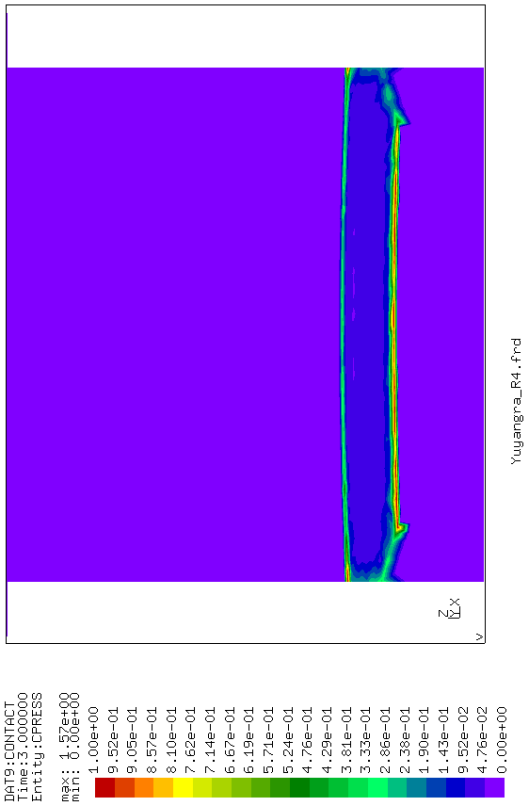
Contact pressure, Nodes.



Contact pressure, LGC7.

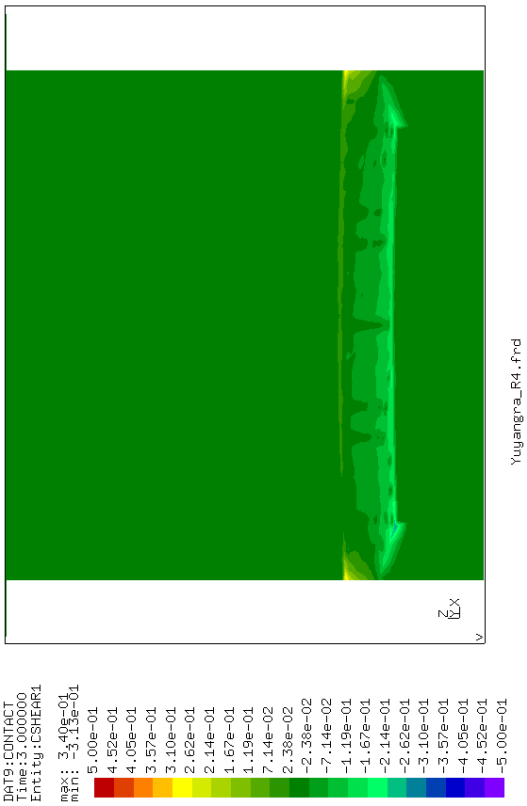


Contact pressure, Abaqus STS.

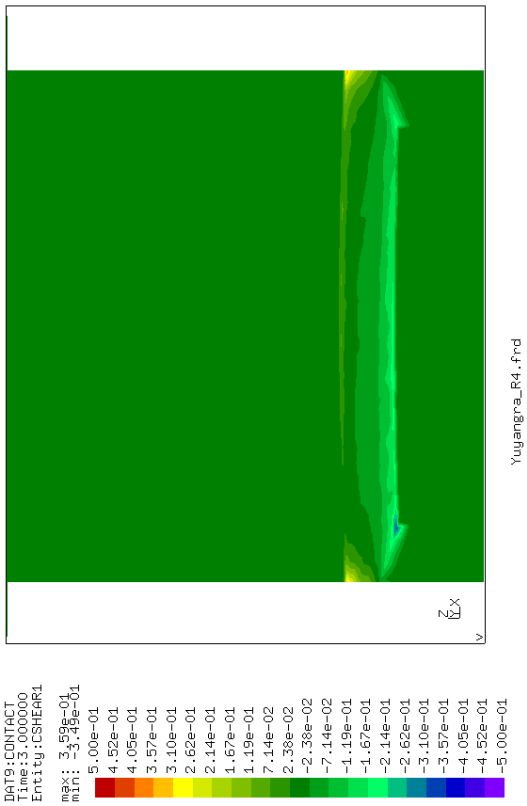


Contact pressure, LGC3.

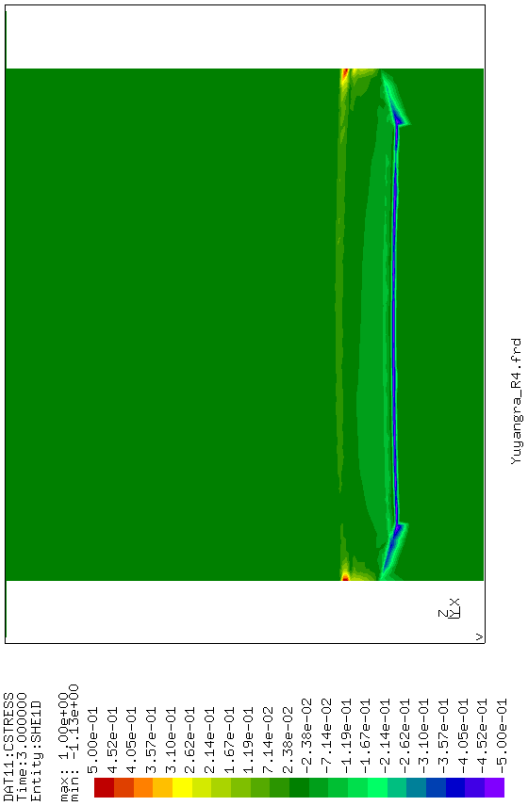
Appendix V: Disk – Blade foot 1 (Y4)



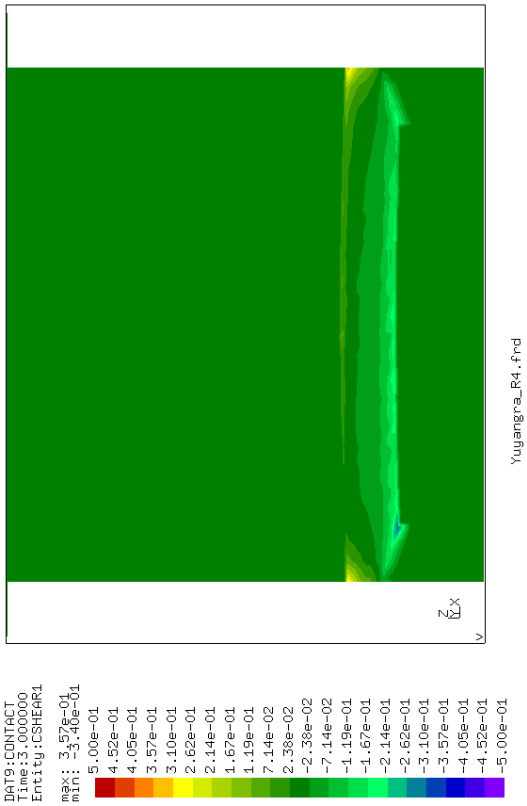
Shear stress (1), Nodes.



Shear stress (1), LGC7.

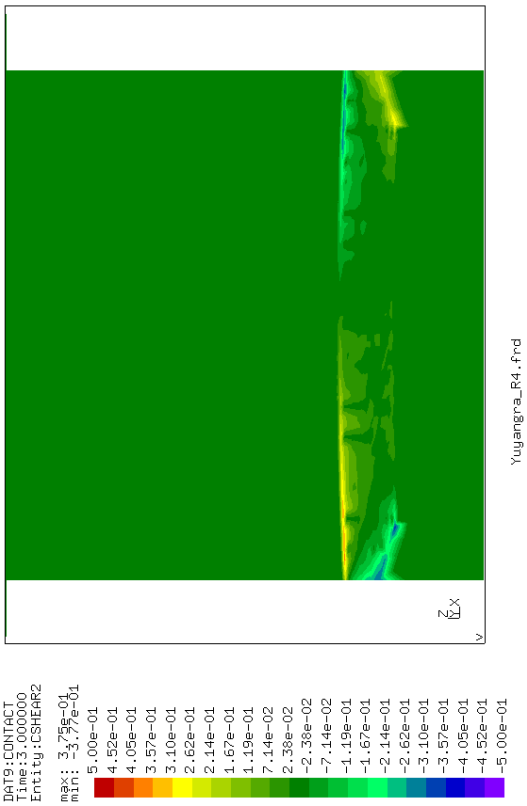


Shear stress (1), Abaqus STS.

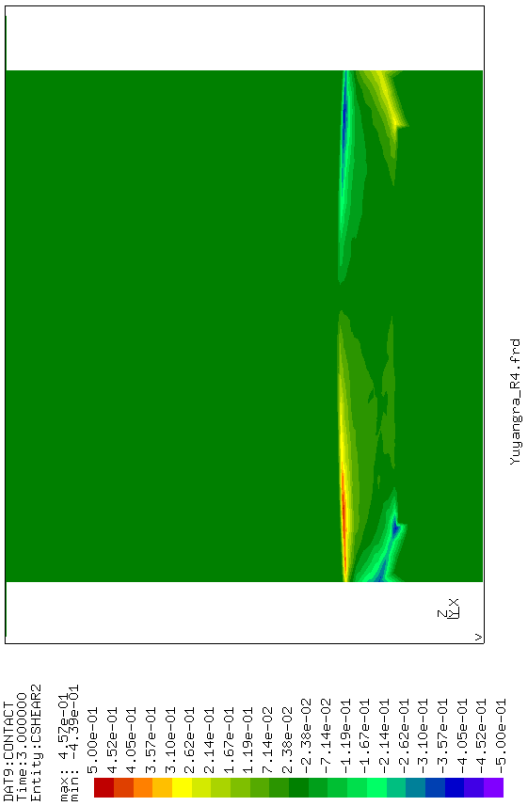


Shear stress (1), LGC3.

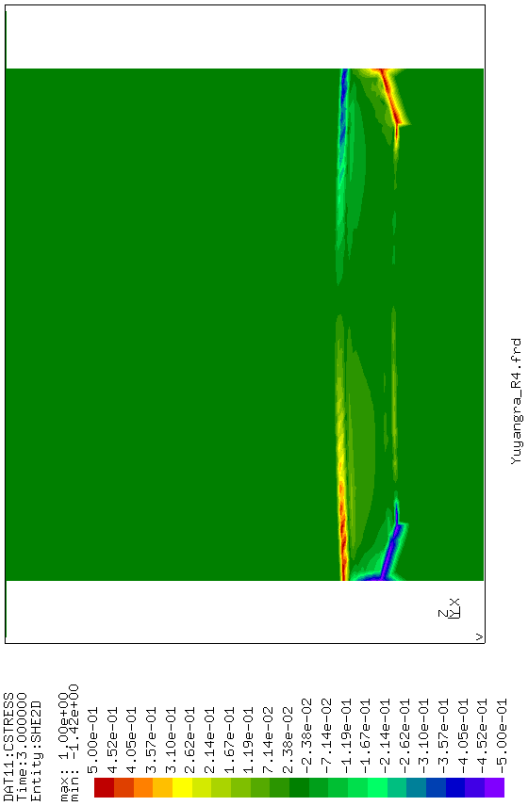
Appendix V: Disk – Blade foot 1 (Y4)



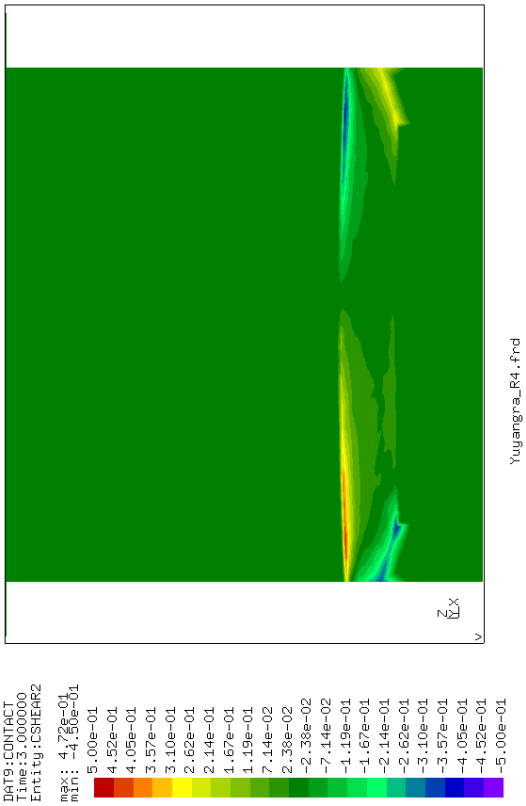
Shear stress (2), Nodes.



Shear stress (2), LGC7.

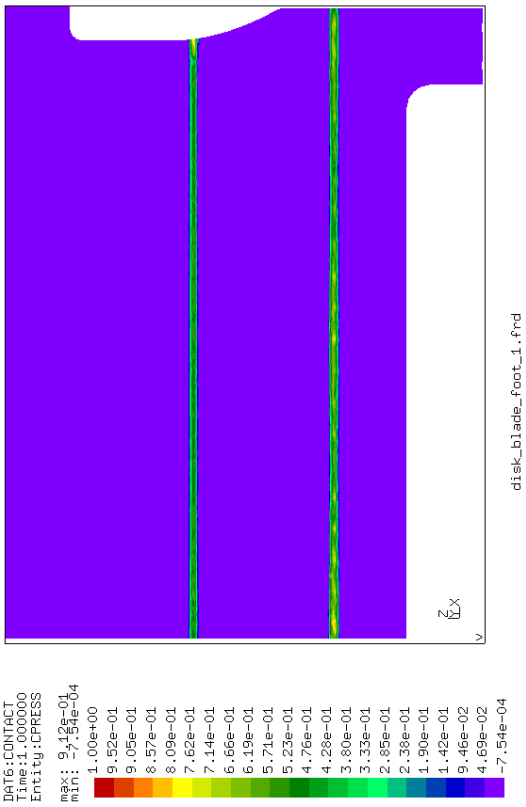


Shear stress (2), Abaqus STS.

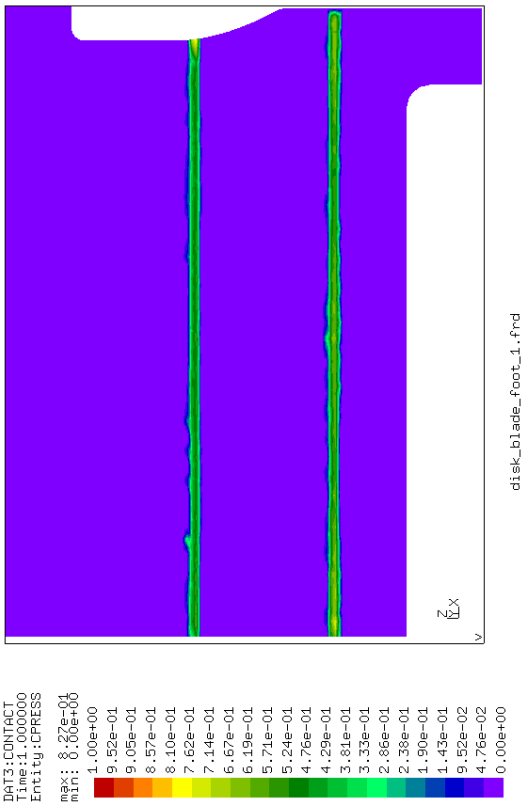


Shear stress (2), LGC3.

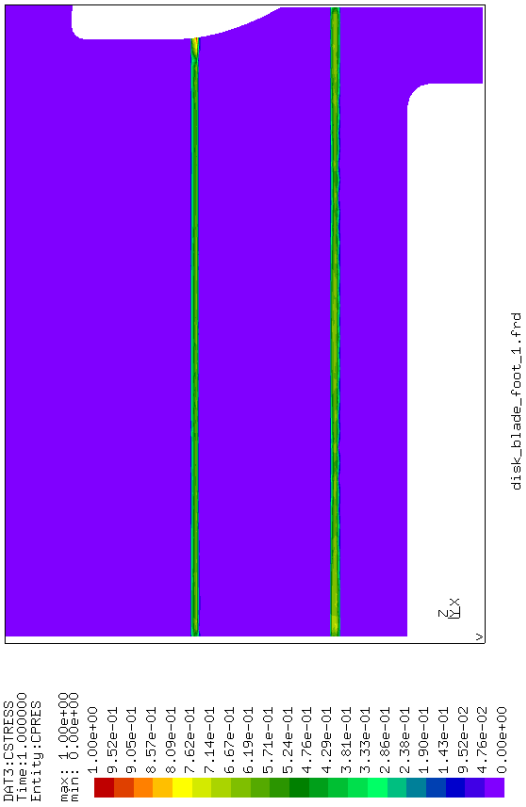
Appendix VI: Disk – Blade foot 2 (DB1)



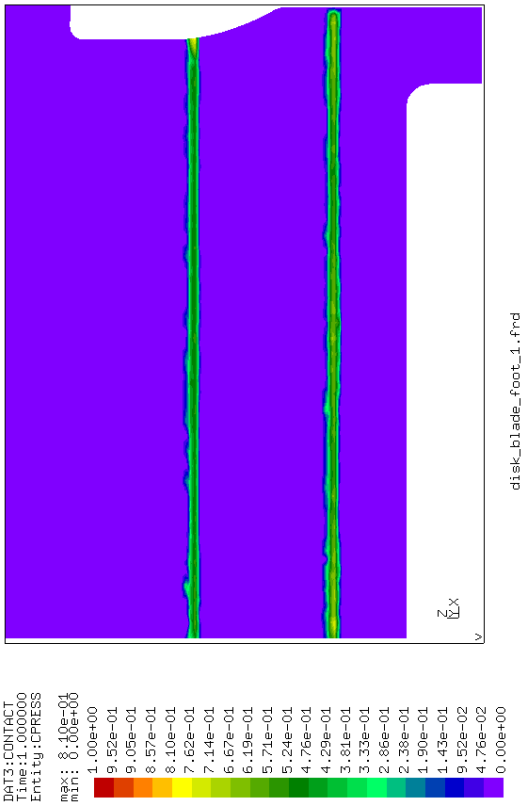
Contact pressure, Nodes.



Contact pressure, LGC7.

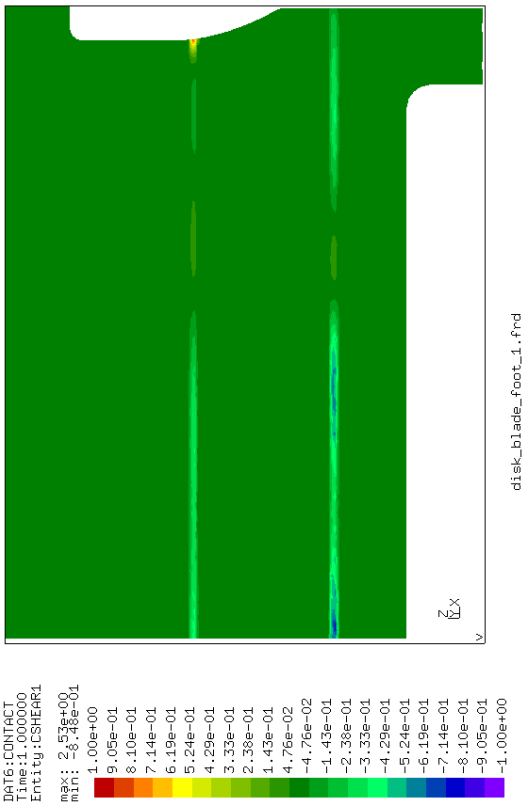


Contact pressure, Abaqus STS.

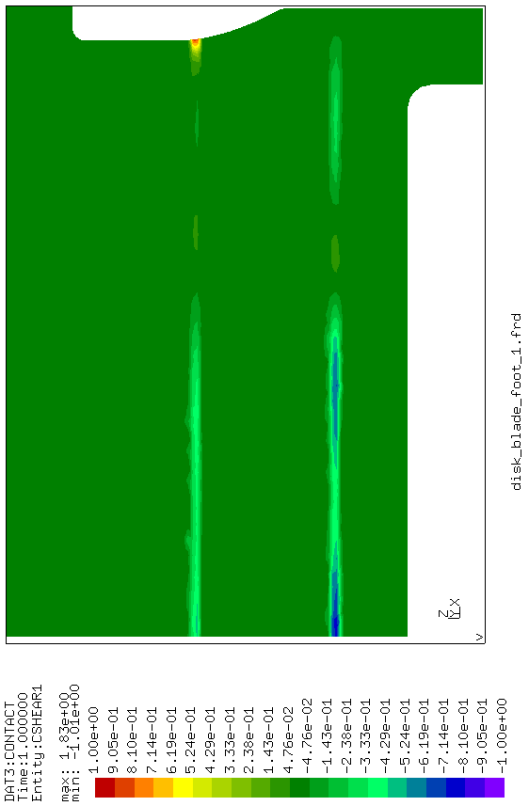


Contact pressure, LGC3.

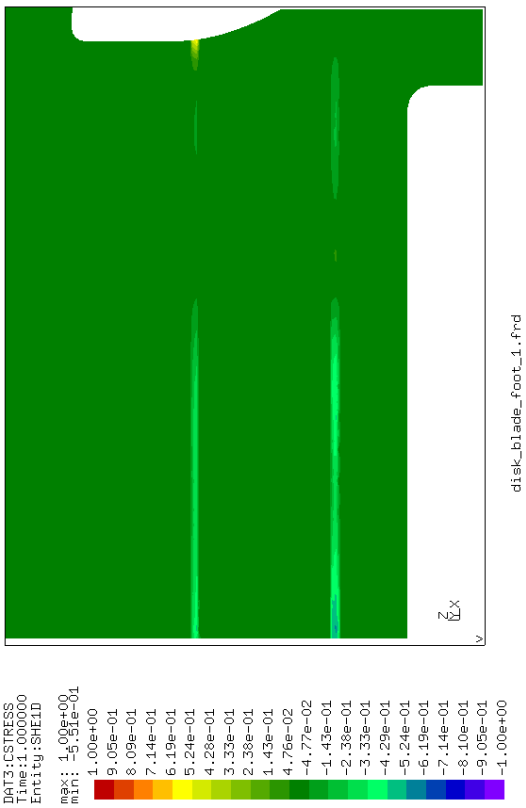
Appendix VI: Disk – Blade foot 2 (DB1)



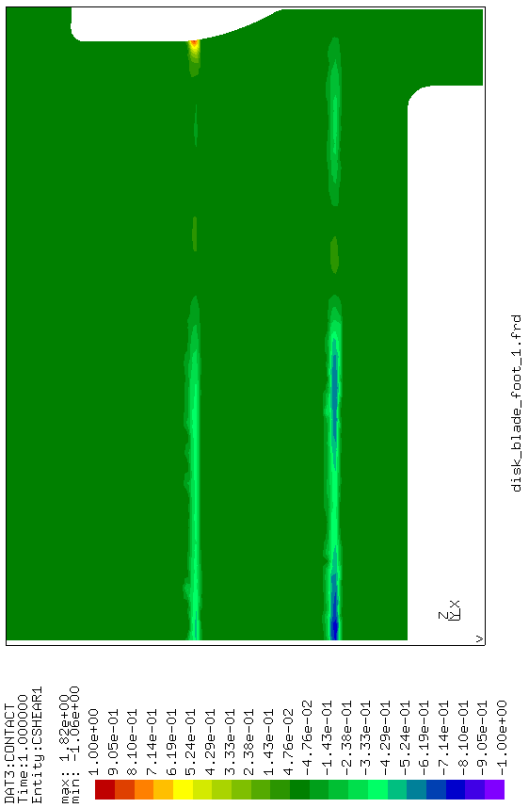
Shear stress (1), Nodes.



Shear stress (1), LGC7.

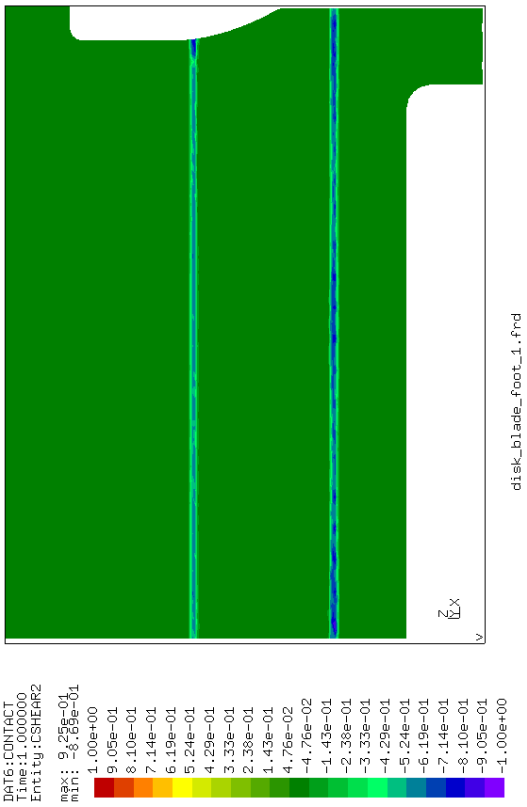


Shear stress (1), Abaqus STS.

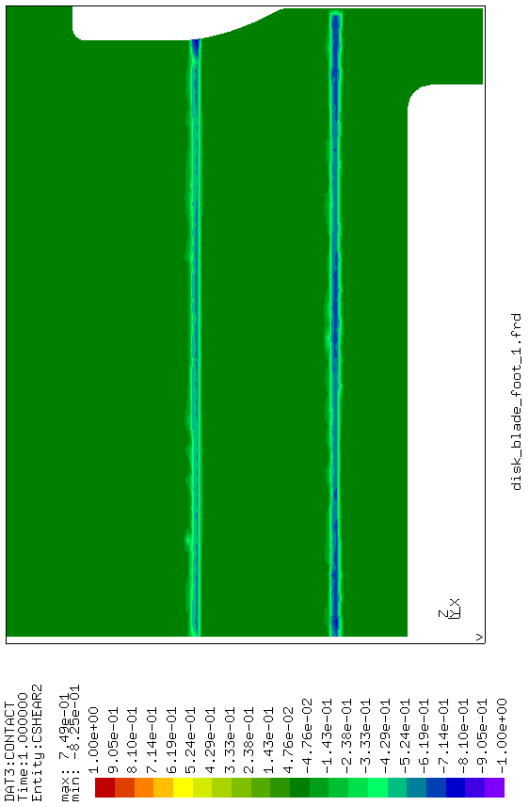


Shear stress (1), LGC3.

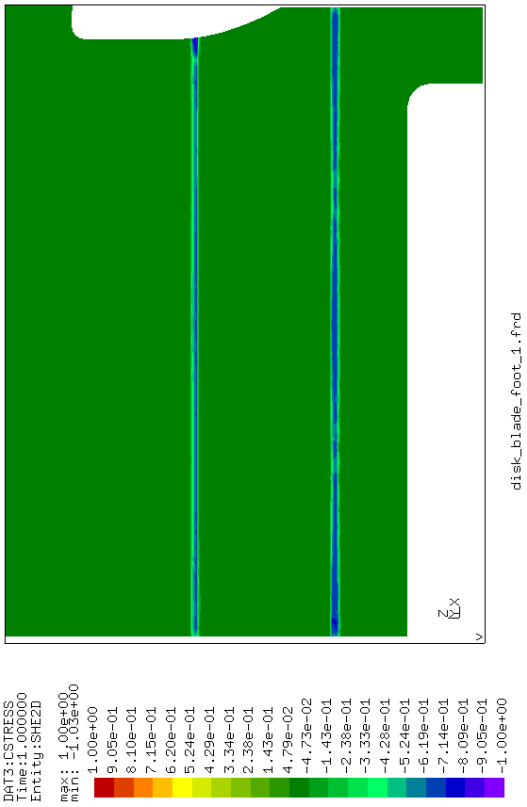
Appendix VI: Disk – Blade foot 2 (DB1)



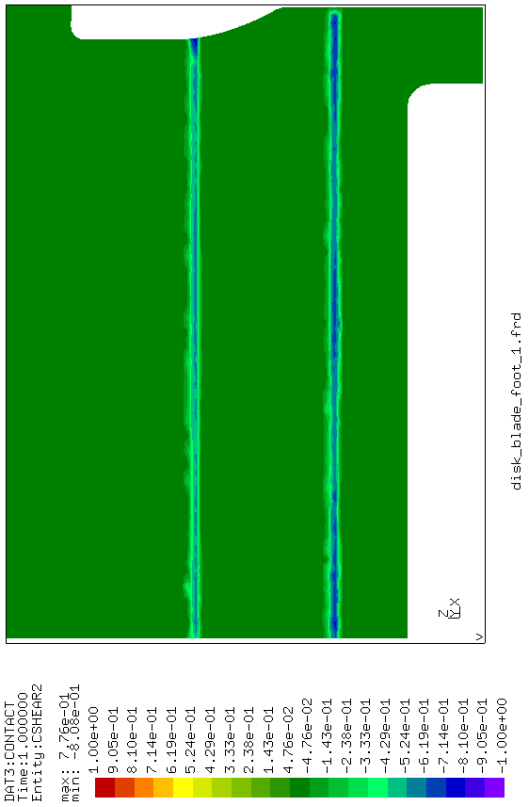
Shear stress (2), Nodes.



Shear stress (2), LGC7.

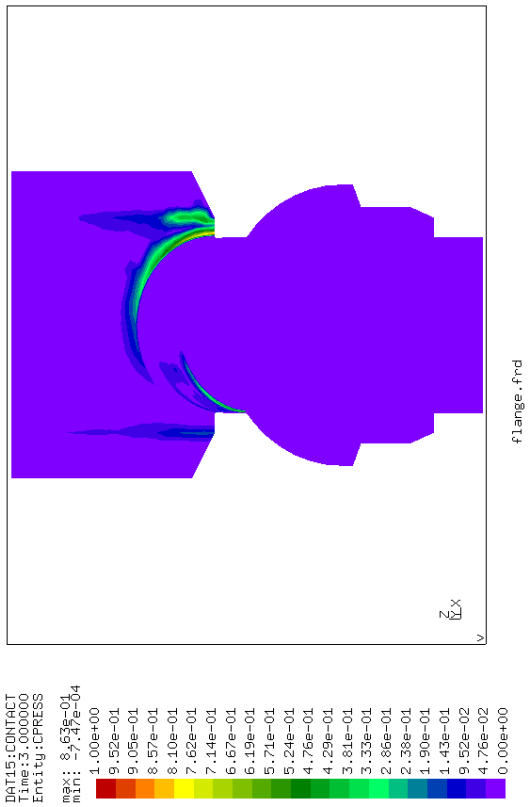


Shear stress (2), Abaqus STS.

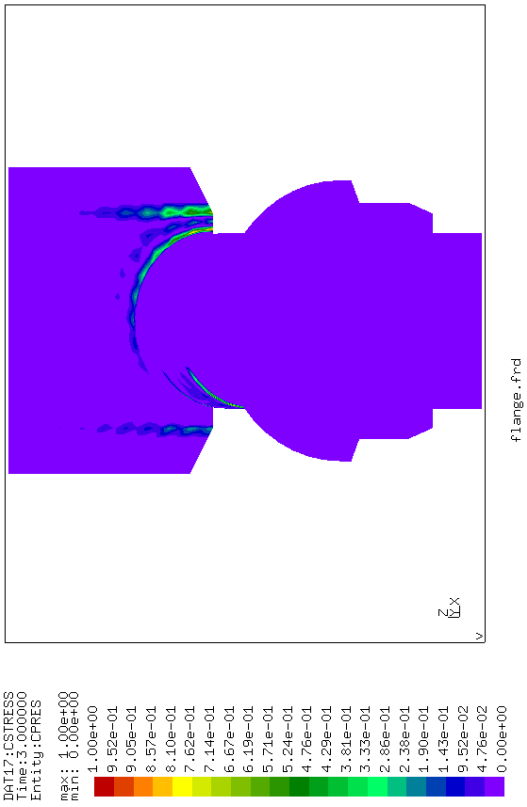


Shear stress (2), LGC3.

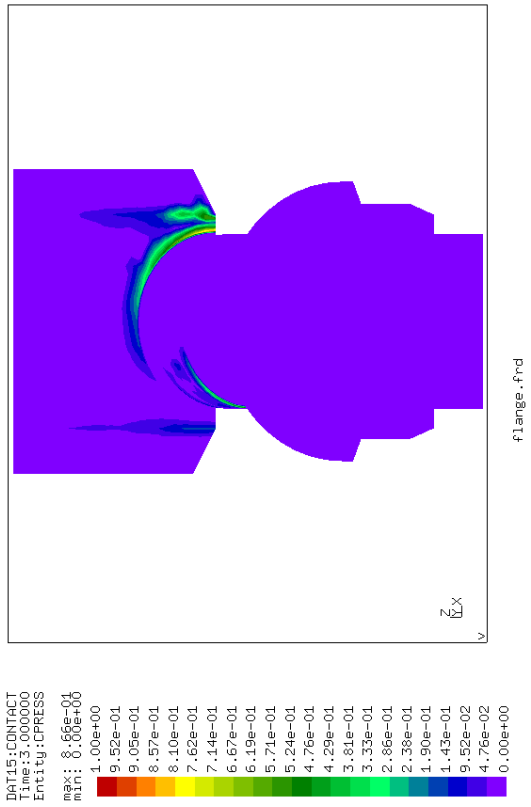
Appendix VII: Flange 1 (FL1)



Contact pressure, LGC7.

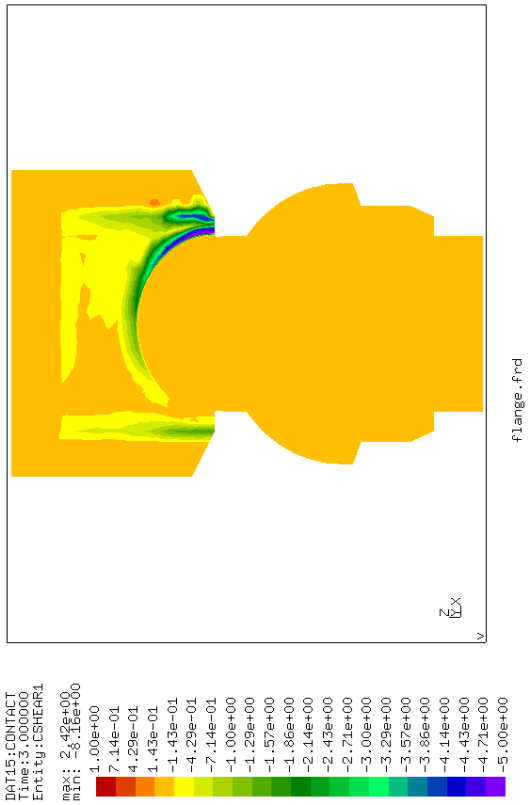


Contact pressure, Abaqus STS.

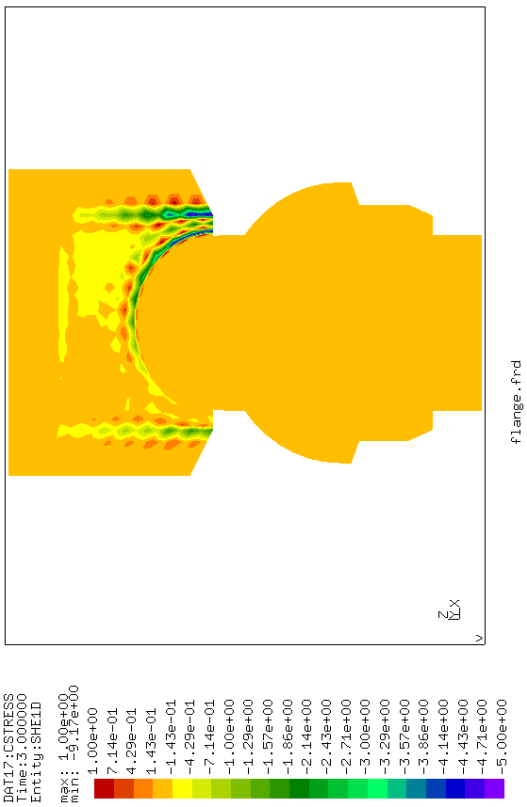


Contact pressure, LGC3.

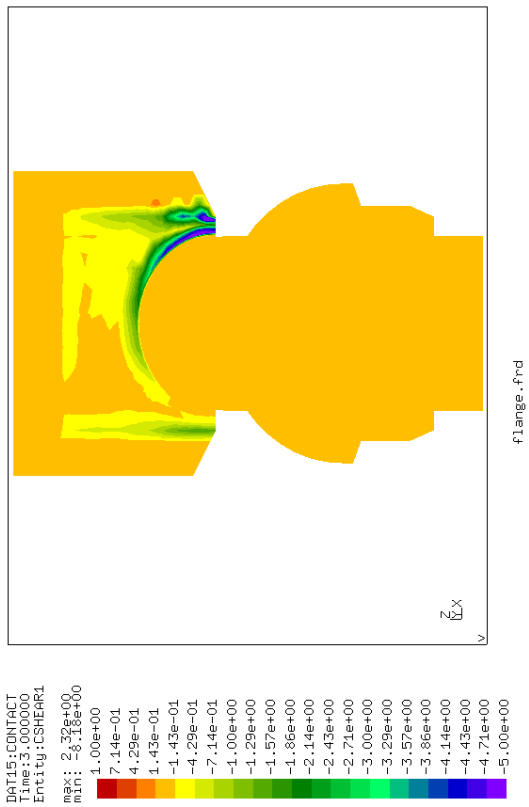
Appendix VII: Flange 1 (FL1)



Shear stress (1), LGC7.

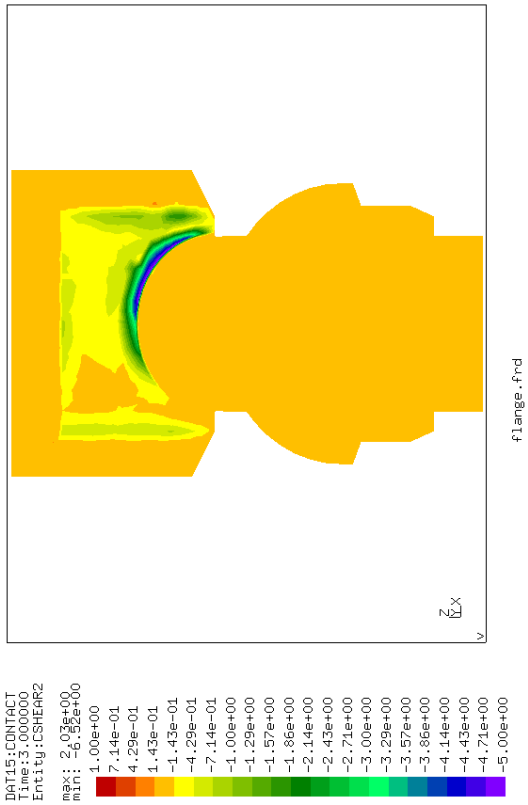


Shear stress (1), Abaqus STS.

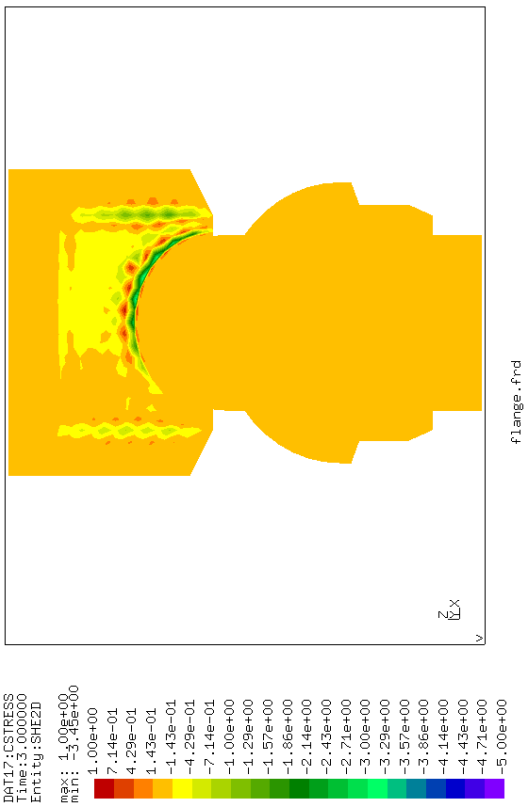


Shear stress (1), LGC3.

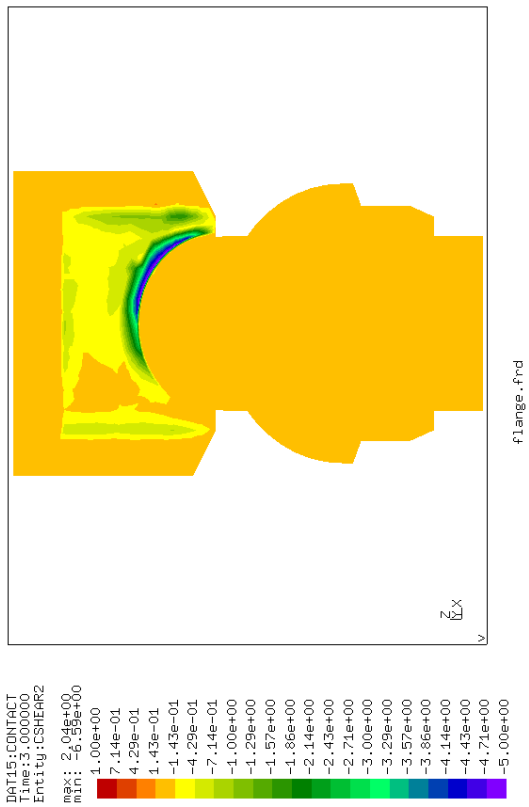
Appendix VII: Flange 1 (FL1)



Shear stress (2), LGC7.

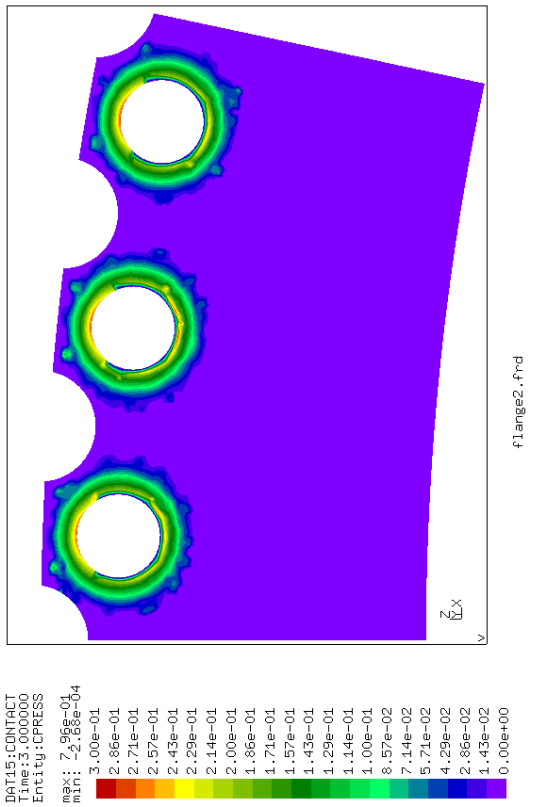


Shear stress (2), Abaqus STS.

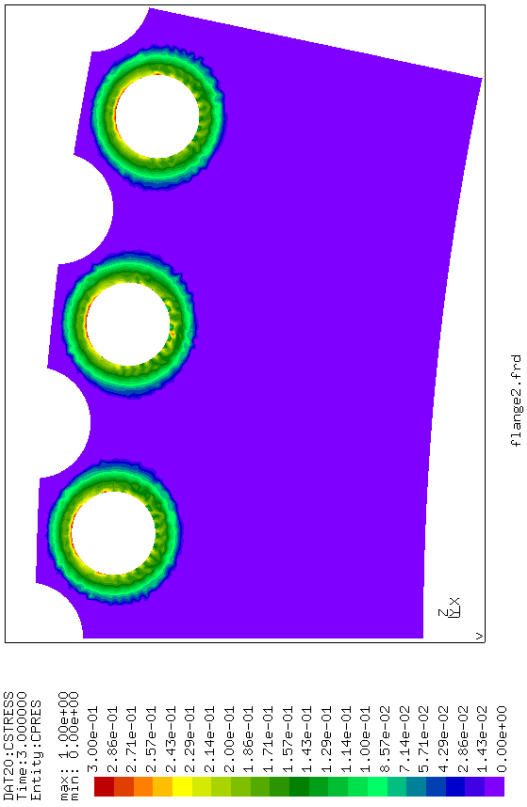


Shear stress (2), LGC3.

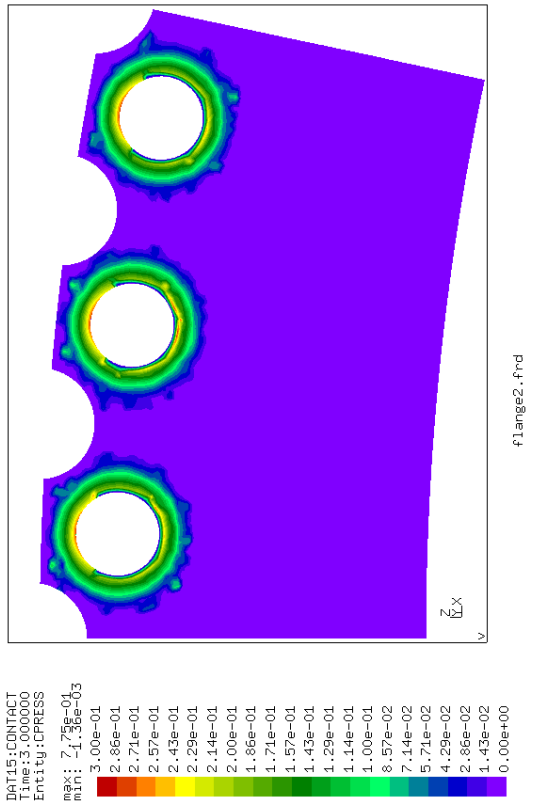
Appendix VIII: Flange 2 (FL2)



Contact pressure, LGC7.

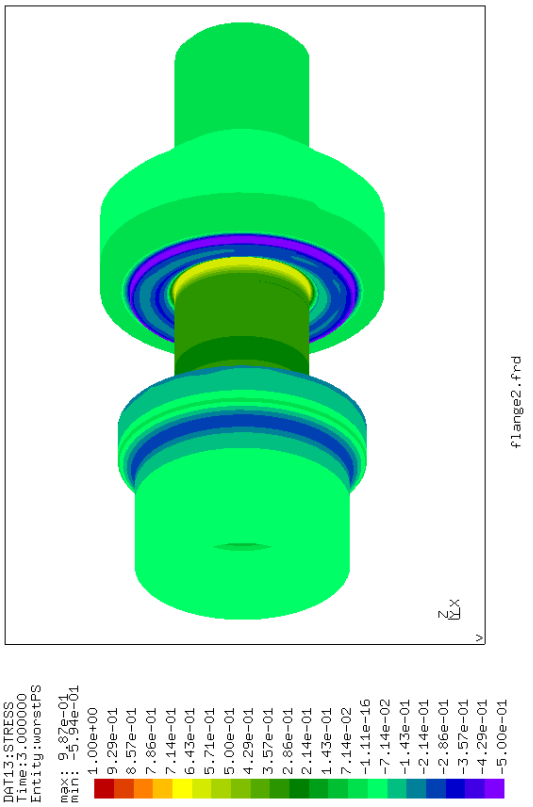


Contact pressure, Abaqus STS.

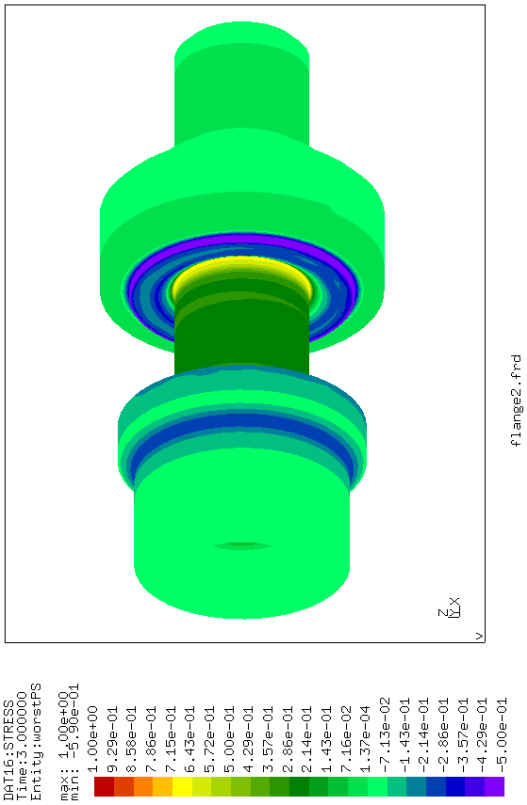


Contact pressure, LGC3.

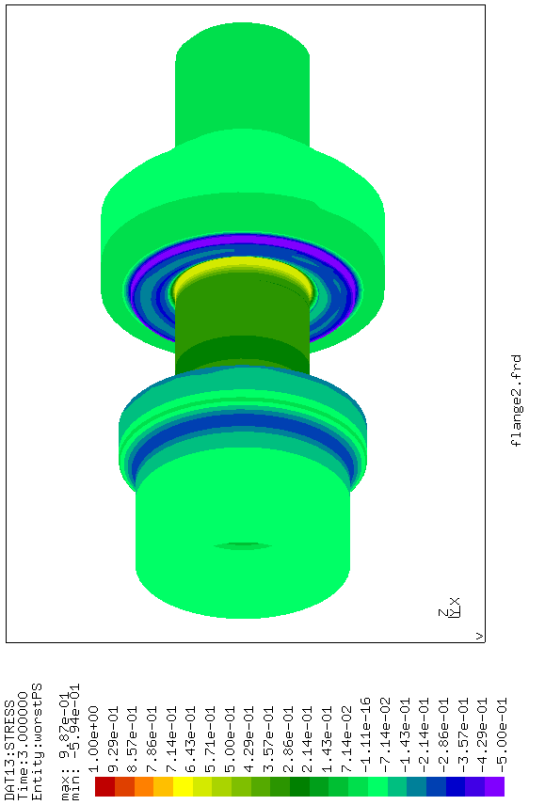
Appendix VIII: Flange 2 (FL2)



Worst principal stress, LGC7.

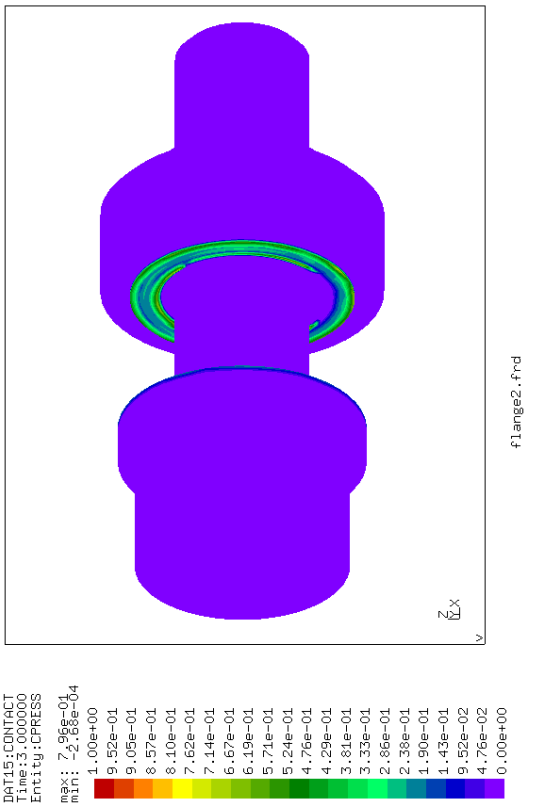


Worst principal stress, Abaqus STS.

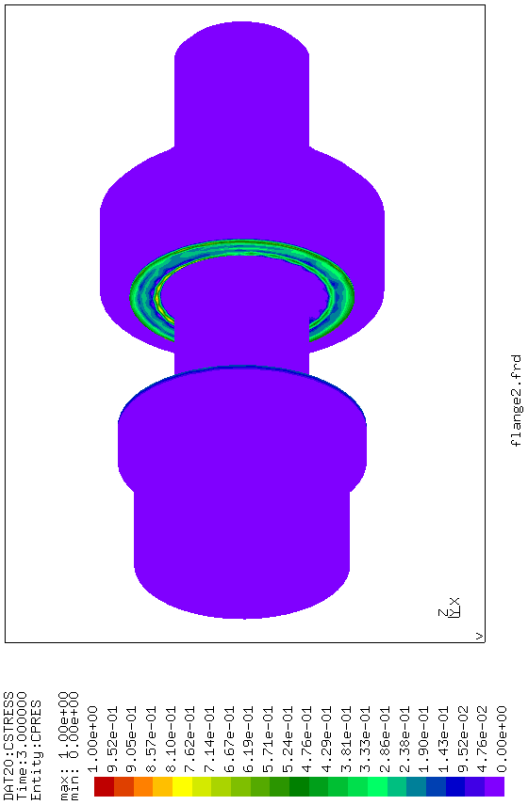


Worst principal stress, LGC3.

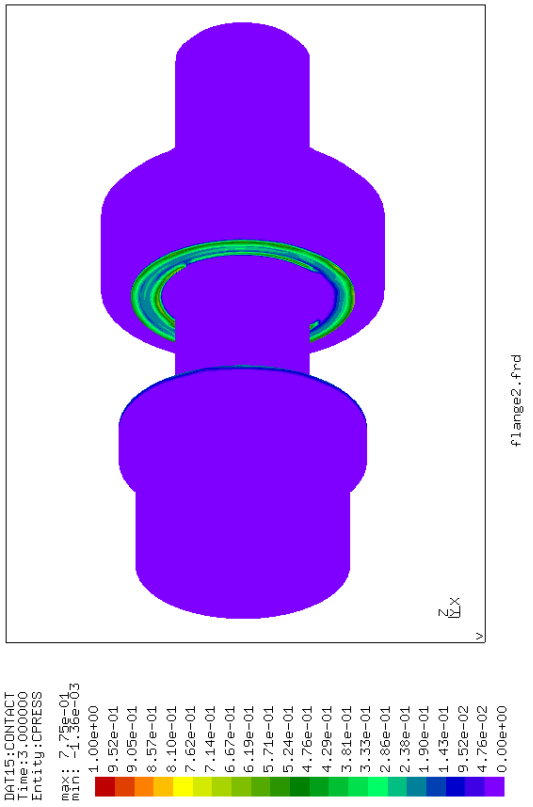
Appendix VIII: Flange 2 (FL2)



Contact pressure, LGC7.

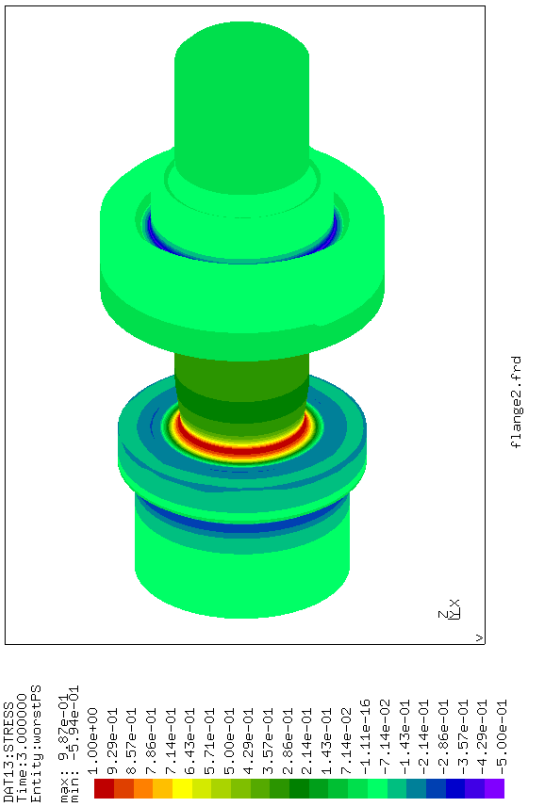


Contact pressure, Abaqus STS.

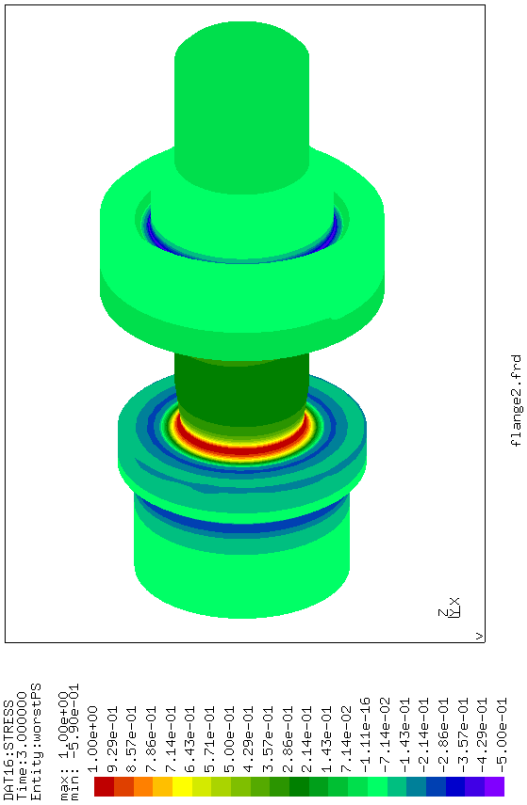


Contact pressure, LGC3.

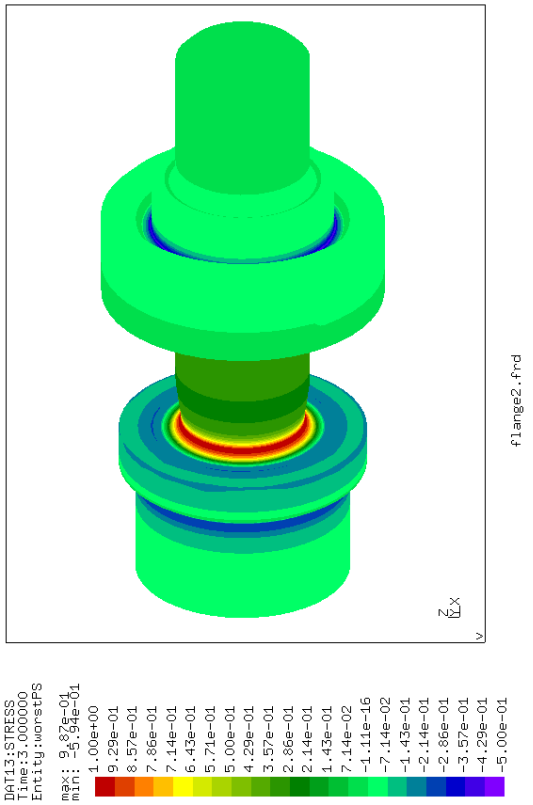
Appendix VIII: Flange 2 (FL2)



Worst principal stress, LGC7.

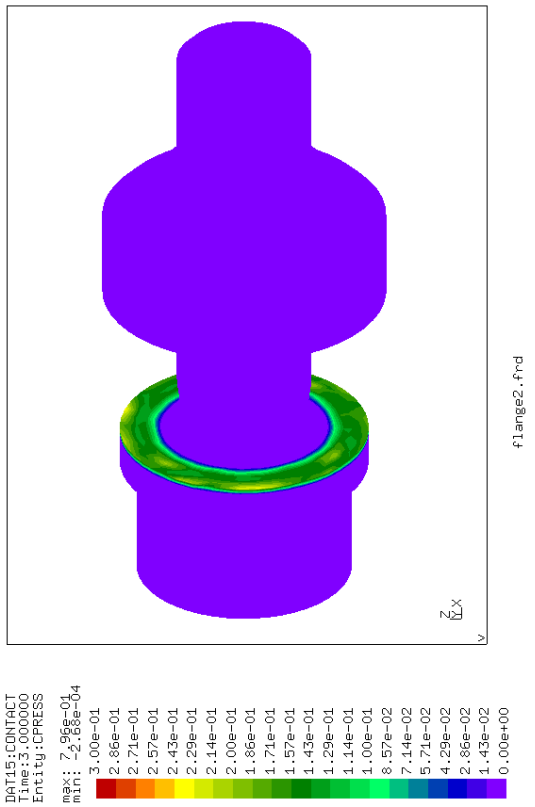


Worst principal stress, Abaqus STS.

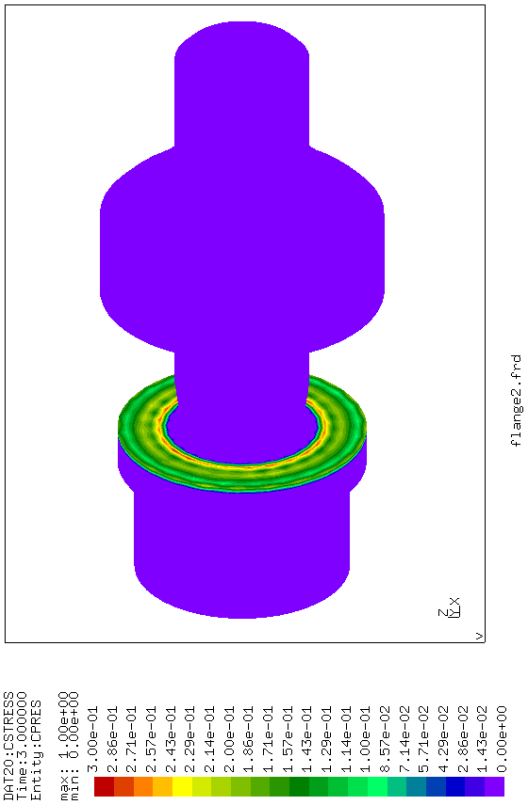


Worst principal stress, LGC3.

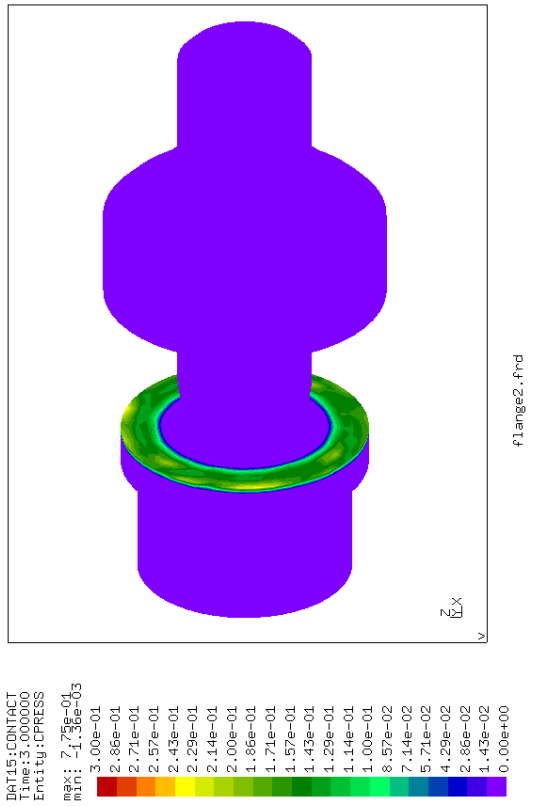
Appendix VIII: Flange 2 (FL2)



Contact pressure, LGC7.

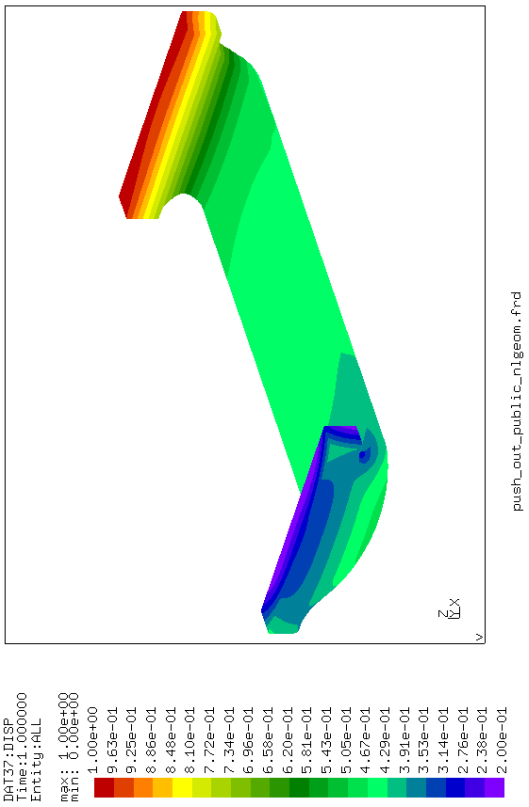


Contact pressure, Abaqus STS.

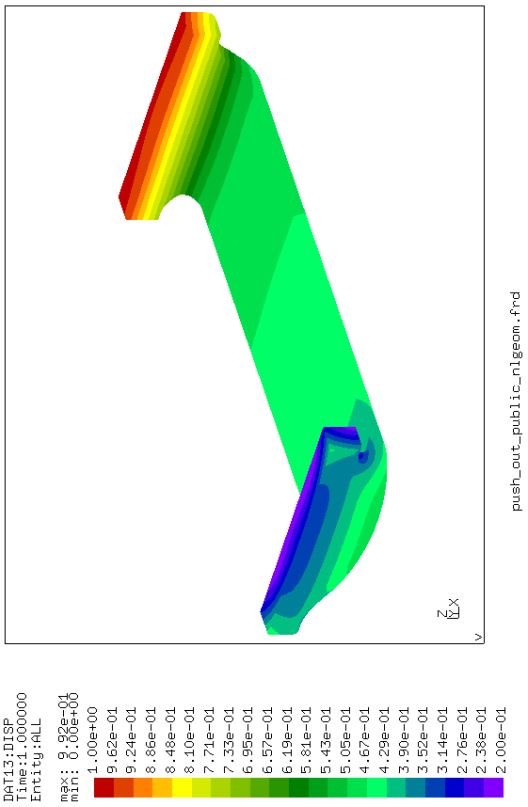


Contact pressure, LGC3.

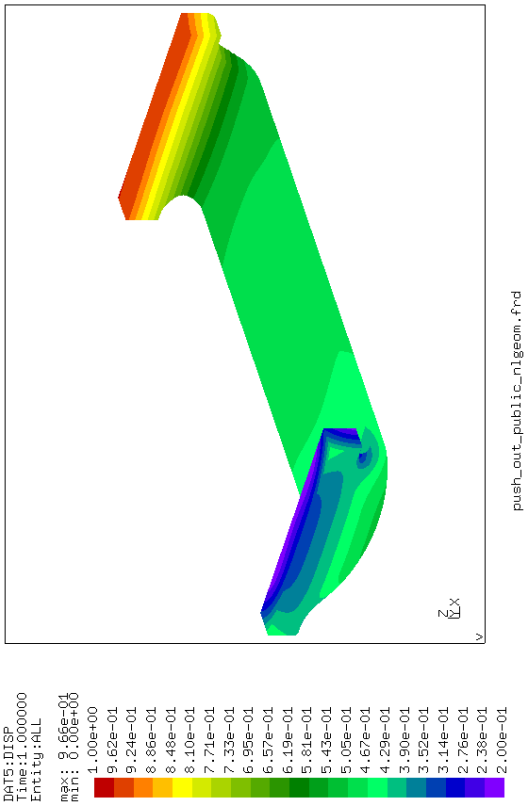
Appendix IX: Push Out Test (PO)



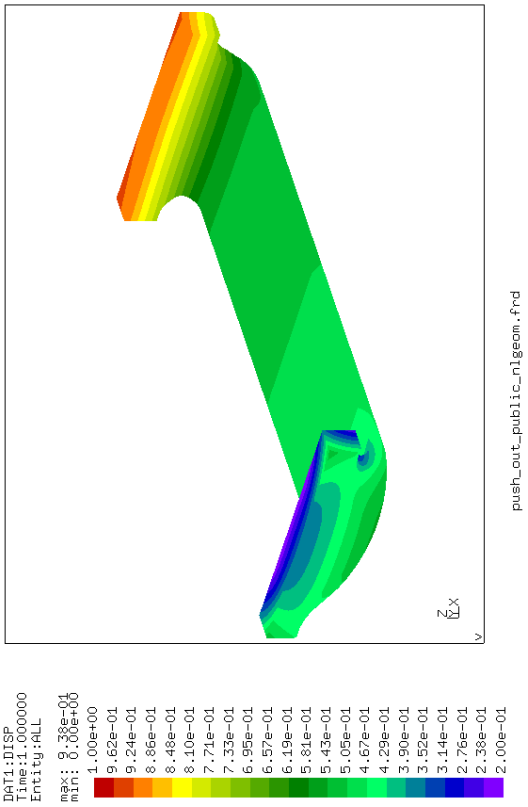
Displacement (INC size: 0.1).



Displacement (INC size: 0.25).

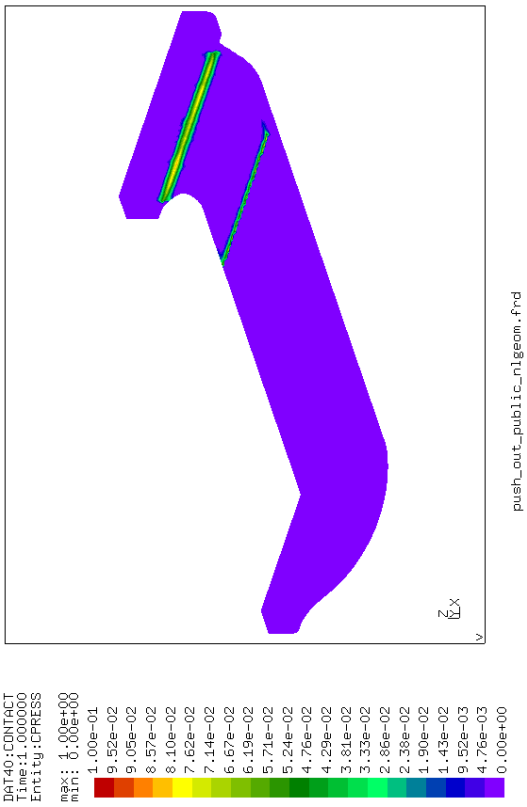


Displacement (INC size: 0.5).

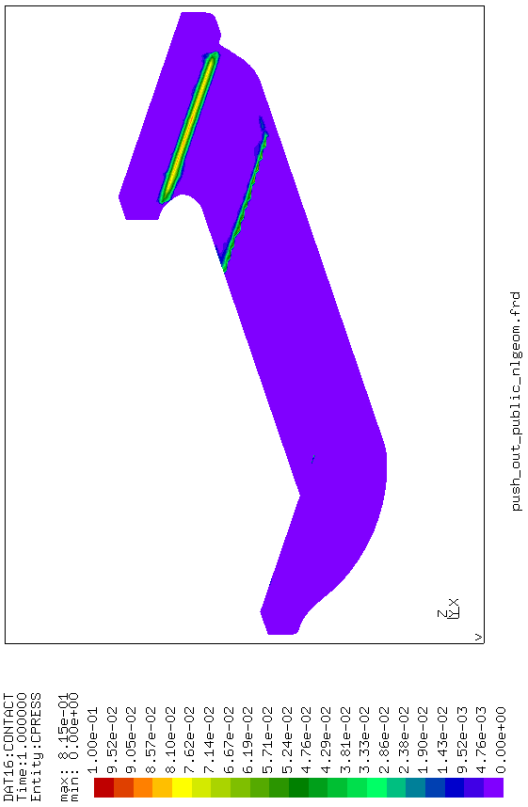


Displacement (INC size: 1).

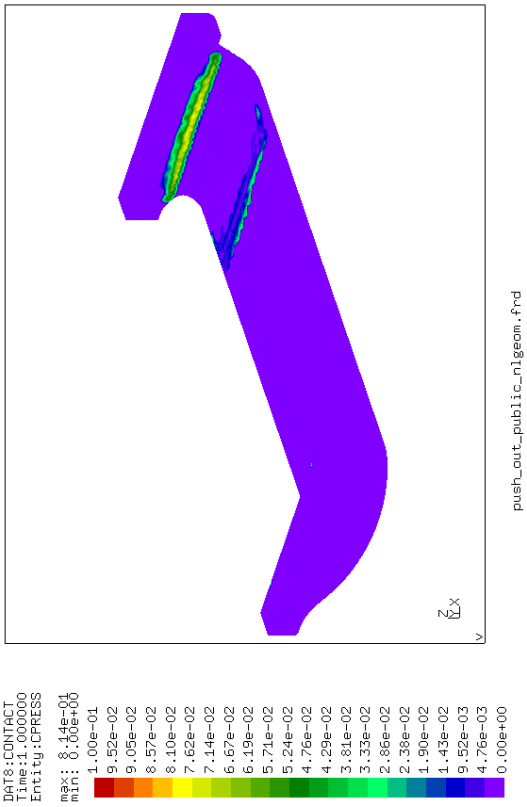
Appendix IX: Push Out Test (PO)



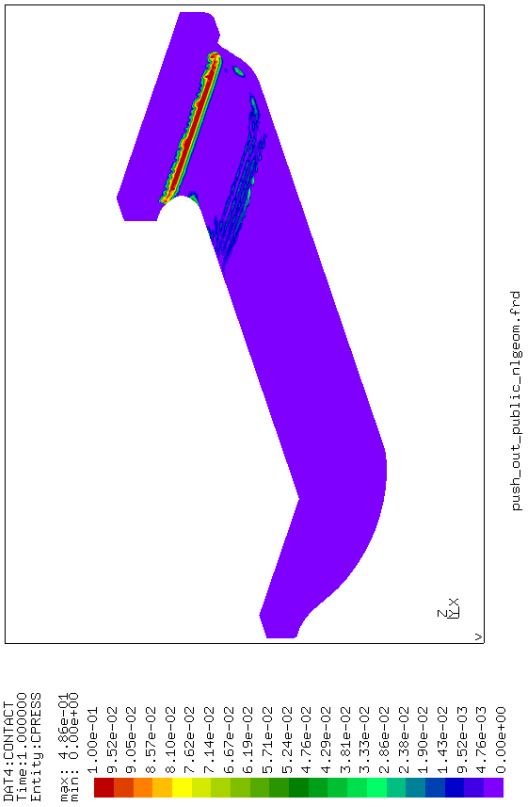
Contact pressure (INC size: 0.1).



Contact pressure (INC size: 0.25).



Contact pressure (INC size: 0.5).



Contact pressure (INC size: 1).